Reţele de calculatoare

EDIŢIA A PATRA

Andrew S. Tanenbaum

Universitatea Vrijie
Amsterdam, Olanda

©2003 Byblos srl, www.byblos.ro

Traducere:

Colectivul de coordonare:

prof. dr. ing. Valentin Cristea

prof. dr. ing. Eugenia Kalisz prof. dr. ing. Nicolae Ţăpuş

Colectivul de traducători:

as.ing. Ana Vârbănescu stud. Corina Stratan prep. ing. Sabina Şerbu ing. Mihaela Negru prep. ing. Natalia Costea as. ing. Răzvan Rughiniş prep. ing. Liviu Dragomirescu stud. Octavian Udrea stud. Bogdan Vişinescu ing. Mihaela Neaţă stud. Vlad Sima stud. Cătălin Cârstoiu stud. Mihai Mircea stud. Cristi Orban stud. Ozana Dragomir stud. Andrei Agapi stud. Ana Maria Oprescu stud. Ionuţ Frujină stud. Gabi Ghiniţă stud. Paul Chiriţă ing. Raluca Busurca stud. Vlad Panait ing. Octavian Purdilă stud. Radu Niculiţă stud. Cătălin Coman

Pregătire, design, producţie:

Mihai Scorţaru, Claudiu Soroiu, Adrian Pop

Editată de BYBLOS s.r.l., ©2003

Bucureşti, Str. Constantin Rădulescu Motru 13/42, Tel: +40-(0)21-3309281

Sub licenţă Pearson Education, Inc. după:

Computer Networks, 4th ed. de *Andrew S. Tanenbaum*

©2003, 1996 by Pearson Education, Inc., Prentice-Hall PTR

Upper Saddle River, New Jersey 07458

Tipărită în România, la MASTER DRUCK,

3400 Cluj-Napoca, Str. Liebknecht 2, Tel: +40-(0)264-432497

ISBN: 973-0-03000-6

Toate drepturile sunt rezervate. Nici o parte a acestei cărţi nu poate fi reprodusă, într-o formă sau printr-un mijloc oarecare, fără permisiunea scrisă a editorului.

Toate numele produselor menţionate aici sunt mărci înregistrate ale respectivilor proprietari.

Reţele de calculatoare

EDIŢIA A PATRA

Pentru Suzanne, Barbara, Marvin şi în memoria lui Bram şi a lui Sweetie n

Alte titluri de mare succes ale lui Andrew S. Tanenbaum:

Sisteme distribuite: principii şi paradigme

Această nouă carte, scrisă împreună cu Maarten van Steen, prezintă atât principiile, cât şi paradigmele sistemelor distribuite moderne. În prima parte sunt tratate în detaliu principiile de comunicare, procesele, numele, sincronizarea, consistenţa şi replicarea, toleranţa la erori şi securitatea. În cea de-a doua parte se trece la prezentarea unor paradigme diferite folosite pentru crearea sistemelor distribuite, inclusiv sisteme bazate pe obiecte, sisteme distribuite de fişiere, sisteme bazate pe documente şi sisteme bazate pe coordonare. Sunt discutate pe larg numeroase exemple.

Sisteme de operare moderne, ediţia a doua

Acest text de mare succes prezintă în detaliu principiile sistemelor de operare şi le ilustrează cu ajutorul a numeroase exemple inspirate din lumea reală. După un prim capitol introductiv, următoarele cinci capitole tratează conceptele de bază: procese şi fire de execuţie, situaţii de blocare, gestiunea memoriei, operaţii de intrare/ieşire. Următoarele şase capitole tratează noţiuni mai avansate, incluzând sisteme multimedia, sisteme multiprocesor, securitate. La sfârşitul cărţii sunt prezentate două studii de caz detaliate: UNIX/Linux şi Windows 2000.

Organizarea structurată a calculatoarelor, ediţia a patra

Această carte clasică, citită în lumea întreagă şi ajunsă acum la cea de-a patra ediţie, furnizează introducerea ideală în studiul arhitecturii calculatoarelor. Subiectul este prezentat într-o manieră uşor de înţeles începând cu prezentarea conceptelor de bază. Există un capitol dedicat începătorilor care prezintă logica digitală, urmat de capitole în care sunt prezentate microarhitectura, setul de instrucţiuni de la nivelul arhitecturii, sistemele de operare, limbajul de asamblare şi arhitecturile paralele de calculatoare.

Sisteme de operare: proiectare şi implementare, ediţia a doua

Acest text despre sisteme de operare, scris împreună cu Albert S. Woodhull, este singura carte ce acoperă atât principiile sistemelor de operare cât şi aplicaţiile

acestora la un sistem real. Sunt tratate în detaliu toate subiectele tradiţionale legate de sistemele de operare. În plus, principiile sunt ilustrate cu grijă de MINIX, un sistem de operare gratuit, de tip UNIX, pentru calculatoare personale. Fiecare carte conţine un CD-ROM care conţine sistemul MINIX complet (cod binar şi sursă). Codul sursă este prezentat într-o anexă a cărţii şi este explicat în detaliu în text.

CUPRINS

PREFAŢĂ **XVI**

1. INTRODUCERE

1.1 UTILIZĂRILE REŢELELOR DE CALCULATOARE 2
1.1.1 Aplicaţii comerciale 3
1.1.2 Aplicaţii domestice 5
1.1.3 Utilizatorii mobili 9
1.1.4 Aspecte sociale 11

1.2 HARDWARE-UL REŢELEI 13
1.2.1 Reţele locale 15
1.2.2 Reţele metropolitane 16
1.2.3 Reţele larg răspândite geografic 17
1.2.4 Reţele fără fir 19
1.2.5 Reţelele casnice (Home networks) 21
1.2.6 Inter-reţelele 23

1.3 PROGRAMELE DE REŢEA 24
1.3.1 Ierarhiile de protocoale 24
1.3.2 Probleme de proiectare a nivelurilor 28
1.3.3 Servicii orientate pe conexiuni şi servicii fără conexiuni 29
1.3.4 Primitive de serviciu 31
1.3.5 Relaţia dintre servicii şi protocoale 33

1.4 MODELE DE REFERINŢĂ 34
1.4.1 Modelul de referinţă OSI 34
1.4.2 Modelul de referinţă TCP/IP 37
1.4.3 O comparaţie între modelele de referinţă OSI şi TCP 40
1.4.4 O critică a modelului şi protocoalelor OSI 41
1.4.5 O critică a modelului de referinţă TCP/IP 43

1.5 EXEMPLE DE REŢELE 44
1.5.1 Internet 44
1.5.5 Reţele orientate pe conexiune 53
1.5.3 Ethernet 59
1.5.4 Reţele fără fir: 802.11 61

1.6 STANDARDIZAREA REŢELELOR 64
1.6.1 Who's Who în lumea telecomunicaţiilor 64
1.6.2 Who's Who în lumea standardelor internaţionale 66
1.6.3 Who's Who în lumea standardelor Internet 68

1.7 UNITĂŢI DE MĂSURĂ 69

1.8 RESTUL CĂRŢII ÎN REZUMAT 70

1.9 REZUMAT 71

1.10 PROBLEME 72

2. NIVELUL FIZIC 77

2.1 BAZELE TEORETICE ALE COMUNICĂRII DE DATE 77
2.1.1 Analiza Fourier 78
2.1.2 Semnalele cu bandă de frecvenţă limitată 78
2.1.3 Viteza maximă de transfer de date a unui canal

81

2.2 MEDII DE TRANSMISIE GHIDATĂ 82
2.2.1 Medii magnetice 82
2.2.2 Cablul torsadat 83
2.2.3 Cablu Coaxial 84
2.2.4 Fibre optice 84

2.3 COMUNICAŢIILE FĂRĂ FIR 90
2.3.1 Spectrul electromagnetic 91
2.3.2 Transmisia radio 93
2.3.3 Transmisia prin microunde 94
2.3.4 Undele infraroşii şi milimetrice 97
2.3.5 Transmisia undelor luminoase 97

2.4 SATELIŢI DE COMUNICAŢIE 98
2.4.1 Sateliţi geostaţionari 99
2.4.2 Sateliţi de altitudine medie 103
2.4.3 Sateliţi de joasă altitudine 103
2.4.4 Sateliţii în comparaţie cu fibrele optice 105

2.5 SISTEMUL TELEFONIC 107
2.5.1 Structura sistemului telefonic 107
2.5.2 Politica din domeniul telefonic 110
2.5.3 Bucla locală: Modemuri, ADSL şi transmisia fără fir 112
2.5.4 Trunchiuri şi multiplexare 123
2.5.5 Comutarea 132

2.6 SISTEMUL DE TELEFONIE MOBILĂ 136
2.6.1 Prima generaţie de telefoane mobile: Voce analogică 137
2.6.2 A doua generaţie de telefoane mobile: Voce digitală 141
2.6.3 A treia generaţie de telefoane mobile: Voce digitală şi date 149

2.7 TELEVIZIUNEA PRIN CABLU 151
2.7.1 Televiziune prin antena colectivă 151
2.7.2 Internet prin cablu 152
2.7.3 Alocarea de spectru 154
2.7.4 Modemuri de cablu 155
2.7.5 Comparaţie între ADSL şi cablu 157

2.8 REZUMAT 158
2.9 PROBLEME 159

3. NIVELUL LEGĂTURĂ DE DATE 165

3.1 ASPECTE ALE PROIECTĂRII NIVELULUI LEGĂTURĂ DE DATE

166

3.1.1 Servicii oferite nivelului reţea 166
3.1.2 Încadrarea 169
3.1.3 Controlul erorilor 172
3.1.4 Controlul fluxului 173

3.2 DETECTAREA ŞI CORECTAREA ERORILOR 173
3.2.1 Coduri corectoare de erori 174
3.2.2 Coduri detectoare de erori 176

3.3 PROTOCOALE ELEMENTARE PENTRU LEGĂTURA DE DATE 179
3.3.1 Un protocol simplex fără restricţii 183
3.3.2 Un protocol simplu Stop-and-Wait (pas-cu-pas) 184
3.3.3 Un protocol simplex pentru un canal cu zgomote 186

3.4 PROTOCOALE CU FEREASTRĂ GLISANTĂ 189
3.4.1 Un protocol cu fereastră glisantă de un bit 191
3.4.2 Un protocol de revenire cu *n* paşi (Go Back n) 194
3.4.3 Un protocol cu repetare selectivă 199

3.5 VERIFICAREA PROTOCOALELOR 204
3.5.1 Modele de tip automat finit 204
3.5.2 Modele de tip reţea Petri 207

3.6 EXEMPLE DE PROTOCOALE ALE LEGĂTURII DE DATE 209
3.6.1 HDLC – Controlul de nivel înalt al legăturii de date 209
3.6.2 Nivelul legăturii de date în Internet 212

3.7 REZUMAT 216

3.8 PROBLEME 217

4. SUBNIVELUL DE ACCES LA MEDIU 22

4.1 PROBLEMA ALOCĂRII CANALULUI 224
4.1.1 Alocarea statică a canalului în reţelele LAN şi MAN 224
4.1.2 Alocarea dinamică a canalului în reţelele LAN şi MAN 225

4.2 PROTOCOALE CU ACCES MULTIPLU 226
4.2.1 ALOHA 226
4.2.2 Protocoale cu acces multiplu şi detecţie de purtătoare 230
4.2.3 Protocoale fără coliziuni 233
4.2.4 Protocoale cu conflict limitat 235
4.2.5 Protocoale cu acces multiplu cu divizarea frecvenţei 238

4.2.6 Protocoale pentru reţele LAN fără fir 241

4.3 ETHERNET 243

4.3.1 Cablarea Ethernet 244
4.3.2 Codificarea Manchester 247
4.3.3 Protocolul subnivelului MAC Ethernet 248
4.3.4 Algoritmul de regresie exponenţială binară 250
4.3.5 Performanţele Ethernet-ului 251
4.3.6 Ethernetul comutat 253
4.3.7 Ethernet-ul rapid 254
4.3.8 Ethernetul Gigabit 257
4.3.9 IEEE 802.2: Controlul legăturilor logice 260
4.3.10 Retrospectiva Ethernetului 261

4.4 REŢELE LOCALE FĂRĂ FIR 262

4.4.1. Stiva de protocoale 802.11 262
4.4.2. Nivelul fizic al 802.11 263
4.4.3 Protocolul subnivelului MAC al 802.11 265
4.4.4 Formatul cadrului 802.11 269
4.4.5 Servicii 270

4.5 REŢELE FĂRĂ FIR DE BANDĂ LARGĂ 271

4.5.1 Comparaţie între 802.11 şi 802.16 272
4.5.2 Stiva de protocoale 802.16 273
4.5.3 Nivelul fizic 802.16 274
4.5.4 Protocolul subnivelului MAC la 802.16 276
4.5.5 Structura cadrului 802.16 278

4.6 BLUETOOTH 278

4.6.1 Arhitectura Bluetooth 279
4.6.2 Aplicaţii Bluetooth 280
4.6.3 Stiva de protocoale Bluetooth 281
4.6.4 Nivelul Bluetooth radio 282
4.6.5 Nivelul bandă de bază Bluetooth 283
4.6.6 Nivelul L2CAP Bluetooth 284
4.6.7 Structura cadrului Bluetooth 284

4.7. COMUTAREA LA NIVELUL LEGĂTURII DE DATE 285

4.7.1 Punţi de la 802.x la 802.y 287
4.7.2 Interconectarea locală a reţelelor 289
4.7.3 Punţi cu arbore de acoperire 290
4.7.4 Punţi aflate la distanţă 292
4.7.5 Repetoare, Noduri, Punţi, Comutatoare, Rutere şi Porţi 292
4.7.6 LAN-uri virtuale 295

4.8 REZUMAT 302

4.9 PROBLEME 303

5. NIVELUL REŢEA 309

5.1 CERINŢELE DE PROIECTARE ALE NIVELULUI REŢEA 309

5.1.1 Comutare de pachete de tip Memorează-şi-Retransmite (Store-and-Forward) 310

5.1.2 Servicii furnizate nivelului transport 310

5.1.3 Implementarea serviciului neorientat pe conexiune 311

5.1.4 Implementarea serviciilor orientate pe conexiune 313

5.1.5 Comparaţie între subreţele cu circuite virtuale şi subreţele datagramă 314

5.2 ALGORITMI DE DIRIJARE 315

5.2.1 Principiul optimalităţii 317

5.2.2 Dirijarea pe calea cea mai scurtă 318

5.2.3 Inundarea 320

5.2.4 Dirijare cu vectori distanţă 321

5.2.5 Dirijarea folosind starea legăturilor 324

5.2.6 Dirijare ierarhică 329

5.2.7 Dirijarea prin difuzare 331

5.2.8 Dirijarea cu trimitere multiplă (multicast) 333

5.2.9 Dirijarea pentru calculatoare gazdă mobile 334

5.2.10 Dirijarea în reţele AD HOC 337

5.2.11 Căutarea nodurilor în reţele punct la punct 341

5.3 ALGORITMI PENTRU CONTROLUL CONGESTIEI 345

5.3.1 Principii generale ale controlului congestiei 347

5.3.2 Politici pentru prevenirea congestiei 348

5.3.3 Controlul congestiei în subreţelele bazate pe circuite virtuale 349

5.3.4 Controlul congestiei în subreţele datagramă 351

5.3.5 Împrăştierea încărcării 353

5.3.6 Controlul fluctuaţiilor 355

5.4 CALITATEA SERVICIILOR 356

5.4.1 Cerinţe 356

5.4.2 Tehnici pentru obţinerea unei bune calităţi a serviciilor 357

5.4.3 Servicii integrate 367

5.4.4 Servicii diferenţiate 370

5.4.5 Comutarea etichetelor şi MPLS 372

5.5 INTERCONECTAREA REŢELELOR 374

5.5.1 Prin ce diferă reţelele 376
5.5.2 Cum pot fi conectate reţelele 377
5.5.3 Circuite virtuale concatenate 378
5.5.4 Interconectarea reţelelor fără conexiuni 379
5.5.5 Trecerea prin tunel 380
5.5.6 Dirijarea în reţele interconectate 382
5.5.7 Fragmentarea 383

5.6 NIVELUL REŢEA ÎN INTERNET 386
5.6.1 Protocolul IP 388
5.6.2 Adrese IP 391
5.5.4 Protocoale de control în Internet 401
5.5.5 Protocolul de dirijare folosit de porţile interioare: OSPF 406
5.6.5 Protocolul de dirijare pentru porţi externe: BGP 411
5.6.6 Trimiterea multiplă în Internet 412
5.6.7 IP mobil 413
5.6.8 IPv6 415

5.7 REZUMAT 423
5.8 PROBLEME 423

6. NIVELUL TRANSPORT 431

6.1 SERVICIILE OFERITE DE NIVELUL TRANSPORT 431
6.1.1 Servicii furnizate nivelurilor superioare 431
6.1.2 Primitivele serviciilor de transport 433
6.1.3 Socluri Berkeley 436
6.1.4 Un exemplu de programare cu socluri: server de fişiere pentru Internet 437

6.2 NOŢIUNI DE BAZĂ DESPRE PROTOCOALELE DE TRANSPORT 441
6.2.1 Adresarea 442
6.2.2 Stabilirea conexiunii 445
6.2.3 Eliberarea conexiunii 449
6.2.4 Controlul fluxului şi memorarea temporară (buffering) 453
6.2.5 Multiplexarea 457
6.2.6 Refacerea după căderе 458

6.3 UN PROTOCOL SIMPLU DE TRANSPORT 460
6.3.1 Primitivele serviciului ales ca exemplu 460
6.3.2 Entitatea de transport aleasă ca exemplu 461
6.3.3 Exemplul văzut ca un automat finit 468

6.4 PROTOCOALE DE TRANSPORT PRIN INTERNET: UDP 471
6.4.1. Introducere în UDP 471
6.4.2. Apel de procedură la distanţă (Remote Procedure Call) 472
6.4.3 Protocolul de transport în timp real – Real-Time Transport Protocol 474

6.5. PROTOCOALE DE TRANSPORT PRIN INTERNET: TCP 477
6.5.1 Introducere în TCP 477
6.5.2 Modelul serviciului TCP 478
6.5.3 Protocolul TCP 480
6.5.4 Antetul segmentului TCP 481
6.5.5 Stabilirea conexiunii TCP 484
6.5.6 Eliberarea conexiunii TCP 485
6.5.7 Modelarea administrării conexiunii TCP 485
6.5.8 Politica TCP de transmisie a datelor 487
6.5.9 Controlul congestiei în TCP 490
6.5.10 Administrarea contorului de timp în TCP 493
6.5.11 TCP şi UDP în conexiune fără fir 496
6.5.12 TCP Tranzacţional 498

6.6 ELEMENTE DE PERFORMANŢĂ 499
6.6.1 Probleme de performanţă în reţelele de calculatoare 500
6.6.2 Măsurarea performanţelor reţelei 502
6.6.3 Proiectarea de sistem pentru performanţe superioare 504
6.6.4 Prelucrarea rapidă a TPDU-urilor 507
6.6.5 Protocoale pentru reţele gigabit 510

6.7 REZUMAT 514

6.8 PROBLEME 515

7. NIVELUL APLICAŢIE 521

7.1 DNS – SISTEMUL NUMELOR DE DOMENII 521
7.1.1 Spaţiul de nume DNS 522
7.1.2 Înregistrări de resurse 524
7.1.3 Servere de nume 527

7.2 POŞTA ELECTRONICĂ 529
7.2.1 Arhitectură şi servicii 530
7.2.2 Agentul utilizator 532
7.2.3 Formatele mesajelor 534
7.2.4 Transferul mesajelor 540
7.2.5 Livrarea finală 543

7.3 WORLD WIDE WEB 548
7.3.1 Aspecte arhitecturale 549
7.3.2 Documente Web statice 564
7.3.3 Documente Web dinamice 576
7.3.4 H'IT'P – HyperText Transfer Protocol 583
7.3.5 Îmbunătăţiri ale performanţei 588
7.3.6 Web-ul fără fir 593

7.4 MULTIMEDIA 602
7.4.1 Introducere în sunetele digitale 603
7.4.2 Compresia audio 605
7.4.3 Fluxuri audio 607
7.4.4 Radio prin Internet 610
7.4.5 Voce peste IP 613
7.4.6 Introducere la video 618
7.4.7 Compresia video 621
7.4.8 Video la cerere 628
7.4.9 MBone – Coloana vertebrală pentru trimitere multiplă 634

7.5 REZUMAT 637

7.6 PROBLEME 638

8. SECURITATEA REŢELELOR 64

8.1 CRIPTOGRAFIA 648
8.1.1 Introducere în criptografie 648
8.1.2 Cifrurile cu substituţie 651
8.1.3 Cifrurile cu transpoziţie 652
8.1.4 Chei acoperitoare 653
8.1.5 Două principii criptografice fundamentale 657

8.2 ALGORITMI CU CHEIE SECRETĂ 658
8.2.1 DES – Data Encryption Standard 660
8.2.2 AES – Advanced Encryption Standard 662
8.2.3 Moduri de cifrare 666
8.2.4 Alte cifruri 670
8.2.5 Criptanaliza 671

8.3 ALGORITMI CU CHEIE PUBLICĂ 671
8.3.1 RSA 672
8.3.2 Alţi algoritmi cu cheie publică 674

8.4 SEMNĂTURI DIGITALE 674

8.4.1 Semnături cu cheie simetrică 675
8.4.2 Semnături cu cheie publică 676
8.4.3 Rezumate de mesaje 677
8.4.4 Atacul zilei de naştere 681

8.5 GESTIONAREA CHEILOR PUBLICE 682
8.5.1 Certificate 683
8.5.2 X.509 684
8.5.3 Infrastructuri cu chei publice 685

8.6 SECURITATEA COMUNICAŢIEI 688
8.6.1 IPsec 689
8.6.2 Ziduri de protecţie 692
8.6.3 Reţele private virtuale 695
8.6.4 Securitatea în comunicaţiile fără fir 696

8.7 PROTOCOALE DE AUTENTIFICARE 700
8.7.1 Autentificare bazată pe cheie secretă partajată 701
8.7.2 Stabilirea unei chei secrete: schimbul de chei Diffie-Hellman 705
8.7.3 Autentificarea folosind un Centru de Distribuţia Cheilor 707
8.7.4 Autentificarea folosind Kerberos 709
8.7.5 Autentificarea folosind criptografia cu cheie publică 711

8.8 CONFIDENŢIALITATEA POŞTEI ELECTRONICE 712
8.8.1 PGP-Pretty Good Privacy (rom.: Confidenţialitate Destul de Bună) 712
8.8.2 PEM-Privacy Enhanced Mail (Poştă cu Confidenţialitate Sporită) 716
8.8.3 S/MIME 717

8.9 SECURITATEA WEB-ULUI 717
8.9.1 Pericole 718
8.9.2 Siguranţa numelor 718
8.9.3 SSL – Nivelul soclurilor sigure (Secure Sockets Layer) 725
8.9.4 Securitatea codului mobil 728

8.10 IMPLICAŢII SOCIALE 730
8.10.1 Confidenţialitate 731
8.10.2 Libertatea de exprimare 733
8.10.3 Dreptul de autor 736

8.11 REZUMAT 738

8.12 PROBLEME 739

9. RECOMANDĂRI DE LECTURĂ ŞI BIBLIOGRAFIE 745

9.1 SUGESTII PENTRU LECTURI VIITOARE 745

9.1.1 Lucrări introductive şi generale 746
9.1.2 Nivelul fizic 747
9.1.3 Nivelul legătură de date 749
9.1.4 Subnivelul de control al accesului la mediu 750
9.1.5 Nivelul reţea 751
9.1.6 Nivelul transport 753
9.1.7 Nivelul aplicaţie 753
9.1.8 Securitatea reţelelor 754

9.2 BIBLIOGRAFIE ÎN ORDINE ALFABETICĂ 756

INDEX 77

PREFAŢĂ

Această carte este acum la a patra ediţie. Fiecare ediţie a corespuns unei etape diferite în modul de utilizare a reţelelor de calculatoare. Când a apărut prima ediţie, în 1980, reţelele erau o curiozitate academică. În 1988, când a apărut a doua ediţie, reţelele erau folosite de universităţi şi de marile firme. Când a apărut a treia ediţie în 1996, reţelele de calculatoare, în special Internet-ul, au devenit o realitate zilnică pentru milioane de oameni. Noutatea celei de a patra ediţii o reprezintă evoluţia rapidă a reţelelor fără fir, în numeroase forme.

Imaginea reţelelor de calculatoare s-a modificat radical de la ediţia a treia. În anii '90 a existat o varietate de reţele de tip LAN şi WAN, împreună cu stivele de protocoale aferente. În anul 2003, singura reţea de tip LAN larg utilizată ce utilizează mediul ghidat de transmisie este Ethernet şi practic toate reţelele WAN existente sunt conectate la Internet. În consecinţă, o importantă cantitate de informaţie referitoare la reţelele mai vechi a fost înlăturată.

Oricum, noile realizări în domeniu sunt şi ele consistente. Cel mai important progres l-au înregistrat comunicaţiile fără fir, inclusiv 802.11, buclele locale de telefonie fără fir, a doua şi a treia generaţie de reţele celulare (2G şi 3G), Bluetooth, WAP, i-mode şi altele. În consecinţă, a fost adăugată o importantă cantitate de informaţie despre reţelele fără fir. Un alt subiect important de actualitate este securitatea în reţele, pentru care a fost adăugat în carte un capitol separat.

Deşi cap. 1 are aceeaşi funcţie introductivă pe care o avea şi în ediţia a treia, cuprinsul a fost revizuit şi actualizat. De exemplu, sunt prezentate introduceri

despre Internet, Ethernet, reţele LAN fără fir, împreună cu istoricul şi originile acestora. Sunt tratate pe scurt şi reţelele pentru utilizatori casnici.

Cap. 2 a fost restructurat într-o oarecare măsură. După o scurtă introducere în principiile comunicaţiilor de date, există trei secţiuni majore despre transmisii (prin medii ghidate, medii fără fir şi sateliţi) urmate de încă trei secţiuni cu studii de caz (reţele comutate de telefonie publică, reţele de telefonie mobilă şi reţele de televiziune prin cablu). Printre noile subiecte expuse în acest capitol se numără ADSL, comunicaţia fără fir în bandă largă, reţele metropolitane fără fir, accesul Internet prin cablu şi DOCSIS.

Cap. 3 s-a ocupat dintotdeauna cu principiile fundamentale ale protocoalelor punct-la-punct. Ideile expuse aici au rămas în vigoare timp de decenii. Ca urmare succesiunea detaliată de exemple de protocoale prezentate în acest capitol a rămas practic neschimbată de la a treia ediţie.

Din contră, în zona subnivelului MAC a existat o activitate intensă în ultimii ani, aşa că s-au produs multe schimbări în cap. 4. Secţiunea dedicată Ethernet-ului a fost extinsă pentru a include şi Gigabit Ethernet. S-au introdus secţiuni complet noi despre LAN-uri fără fir, comunicaţie fără fir în bandă largă, Bluetooth şi comutare la nivel de legătură de date, inclusiv MPLS.

Cap. 5 a fost de asemenea actualizat: au fost înlăturate toate materialele referitoare la ATM şi au fost adăugate materiale suplimentare despre Internet. Un alt subiect important este calitatea serviciilor, cuprinzând expuneri despre servicii integrate şi servicii diferenţiate. Sunt prezente în acest capitol şi reţelele fără fir, împreună cu o discuţie despre rutarea în reţele ad-hoc. Alte aspecte noi includ NAT şi reţelele de la egal la egal (peer-to-peer).

Cap. 6 tratează în continuare nivelul transport, dar şi aici au avut loc unele schimbări. Printre acestea se numără un exemplu de programare a soclurilor (sockets). Sunt prezentate şi comentate două programe de câte o pagină scrise în limbajul C. Aceste programe, disponibile şi pe situl Web al cărţii, pot fi compilate şi rulate. Împreună ele furnizează o aplicaţie de server de fişiere sau server de Web, pentru experimentare. Alte subiecte noi includ apelul procedurilor la distanţă, RTP şi tranzacţii/TCP.

Cap. 7, relativ la nivelul aplicaţie, a fost mai clar orientat. După o scurtă introducere în DNS, restul capitolului tratează trei aspecte: poştă electronică, Web şi multimedia. Fiecare dintre acestea este tratată foarte detaliat. Discuţia despre modul de funcţionare a Web-ului se întinde acum pe mai mult de 60 de pagini, acoperind multe subiecte, printre care pagini Web statice şi dinamice, HTTP, scripturi CGI, reţele cu livrare bazată pe conţinut, cookies şi păstrarea temporară în memoria ascunsă (cache) a Web-ului. Sunt prezente şi materiale despre modul de scriere a paginilor Web moderne, cu scurte introduceri în XML, XSL, XHTML, PHP şi altele, toate însoţite de exemple funcţionale. Este menţionat şi accesul Web fără

fir, cu accent asupra i-mode şi WAP. Secţiunea de multimedia cuprinde acum MP3, fluxuri audio, radio prin Internet şi transmisii de voce peste IP.

Securitatea reţelelor a devenit azi atât de importantă încât i s-a acordat un nou capitol însumând peste 100 de pagini. Sunt prezentate atât principii de securitate (algoritmi simetrici şi algoritmi cu chei publice, semnături digitale şi certificate X.509) cât şi aplicaţii ale acestor principii (autentificare, securitatea poştei electronice şi securitatea Web). Acest capitol este atât întins ca arie de acoperire (de la criptografie cuantică până la cenzura guvernamentală) cât şi bogat în detalii (de exemplu modul de funcţionare al algoritmului SHA-1).

Cap. 9, conţine o listă complet nouă de recomandări bibliografice, cât şi o bibliografie cuprinzătoare de peste 350 de titluri. Peste 200 dintre aceste lucrări sunt scrise după anul 2000.

Cărţile despre computere conţin foarte multe acronime. Nici cartea de faţă nu face excepţie. După ce aţi terminat de citit această carte, următorii termeni ar trebui să însemne ceva pentru dumneavoastră: ADSL, AES, AMPS, AODV, ARP, ATM, BGP, CDMA, CDN, CGI, CIDR, DCF, DES, DHCP, DMCA, FDM, FHSS, GPRS, GSM, HDLC, HFC, HTML, HTTP, ICMP, IMAP, ISP, ITU, LAN, LMDS, MAC, MACA, MIME, MPEG, MPLS, MTU, NAP, NAT, NSA, NTSC, OFDM, OSPF, PCF, PCM, PGP, PHP, PKI, POTS, PPP, PSTN, QAM, QPSK, RED, RFC, RSA, RSVP, RTP, SSL, TCP, TDM, UDP, URL, UTP, VLAN, VPN, VSAT, WAN, WAP, WDMA, WEP, WWW şi XML. Dar nu vă îngrijoraţi. Ficare din aceşti termeni va fi cu atenţie explicat înainte de a fi utilizat. Pentru a-i ajuta pe instructorii care doresc să folosească această carte ca suport de curs, autorul a pregătit o varietate de materiale auxiliare, printre care:

Un manual cu soluţiile problemelor.
Fişiere conţinând toate figurile în diferite formate
Un simulator (scris în C) pentru exemplele de protocoale din Cap. 3.
O pagină de web cu link-uri către îndrumare practice, organizaţii, întrebări frecvente, etc

Manualul cu soluţii este disponibil la Prentice Hall (dar numai pentru instructori, nu şi pentru stu- denţi). Toate celelalte materiale pot fi găsite pe situl cărţii, la adresa:

http:llwww.prenhall.comltanenbaum

De acolo, faceţi click pe coperta cărţii.

Multe persoane m-au ajutat în timpul lucrului la a patra ediţie. Aş dori în mod deosebit să mulţumesc următoarelor persoane: Ross Anderson, Elizabeth Belding-Royer, Steve Bellovin, Chatschick Bisdikian, Kees Bot, Scott Bradner, Jennifer Bray, Pat Cain, Ed Felten, Warwick Ford, Kevin Fu, Ron Fulle, Jim Geier, Mario Gerla, Natalie Giroux, Steve Hanna, Jeff Hayes, Amir Herzberg, Philip Homburg, Philipp

Hoschka, David Green, Bart Jacobs, Frans Kaashoek, Steve Kent, Roger Kermode, Robert Kinicki, Shay Kutten, Rob Lanphier, Marcus Leech, Tom Maufer, Brent Miller, Shivakant Mishra, Thomas Nadeau, Shlomo Ovadia, Kaveh Pahlavan, Radia Perlman, Guillaume Pierre, Wayne Pleasant, Patrick Powell, Thomas Robertazzi, Medy Sanadidi, Christian Schmutzer, Henning Schulzrinne, Paul Sevinc, Mihail Sichitiu, Bernard Sklar, Ed Skoudis, Bob Strader, George Swallow, George Thiruvathukal, Peter Tomsu, Patrick Verkaik, Dave Vittali, Spyros Voulgaris, Jan-Mark Wams, Ruediger Weis, Bert Wijnen, Joseph Wilkes, Leendert van Doorn şi Maarten van Steen.

Mulţumiri speciale sunt adresate lui Trudy Levine care a demonstrat că bunicile pot face o treabă excelentă recapitulând materialul tehnic. Shivakant Mishra s-a gândit la multe dintre problemele dificile de la sfârşitul capitolelor. Andy Dornan mi-a recomandat lecturi suplimentare pentru Cap. 9. Jan Looyen a furnizat echipamente hardware indispensabile într-un moment critic. Dr. F de Nies s-a dovedit un expert în materie de ”cut-and-paste” atunci când a fost necesar. Editorul meu de la Prentice Hall, Mary Franz m-a aprovizionat cu mai multe materiale pentru citit decât consumasem în precedenţii 7 ani şi m-a ajutat în numeroase alte situaţii.

În sfârşit, am ajuns la persoanele cele mai importante: Suzanne, Barbara şi Marvin. Suzannei pentru dragoste, răbdare şi prânzurile din excursiile la iarbă verde. Barbarei şi lui Marvin pentru că au fost veseli şi amuzanţi în permanenţă (mai puţin atunci cand se plângeau de îngrozitoarele manuale pentru colegiu, fapt ce m-a făcut să fiu mai cu picioarele pe pământ). Vă mulţumesc.

ANDREW S. TANENBAUM

INTRODUCERE

Fiecare din ultimele trei secole a fost dominat de o anumită tehnologie. Secolul al XVIII-lea a fost secolul marilor sisteme mecanice care au însoţit Revoluţia Industrială. Secolul al XIX-lea a fost epoca maşinilor cu aburi. În secolul XX, tehnologia cheie este legată de colectarea, prelucrarea şi distribuirea informaţiei. Printre alte realizări, am asistat la instalarea reţelelor telefonice mondiale, la invenţia radioului şi a televiziunii, la naşterea şi creşterea nemaivăzută a industriei de calculatoare şi la lansarea sateliţilor de comunicaţii.

Datorită progresului tehnologic rapid, aceste domenii converg în ritm alert, iar diferenţele între colectarea, transportul, stocarea şi prelucrarea informaţiei dispar pe zi ce trece. Organizaţii cu sute de birouri răspândite pe o arie geografică largă se aşteaptă să poată examina în mod curent printr-o simplă apăsare de buton chiar şi echipamentele lor cele mai îndepărtate. Pe măsură ce posibilităţile noastre de a colecta, prelucra şi distribui informaţia cresc tot mai mult, cererea pentru o prelucrarea şi mai sofisticată a informaţiei creşte şi mai rapid.

Deşi industria de calculatoare este încă tânără în comparaţie cu alte industrii (de exemplu, construcţia de automobile şi transportul aerian), domeniul calculatoarelor a cunoscut un progres spectaculos într-un timp scurt. În primele decenii de existenţă sistemele de calcul erau foarte centralizate, de obicei în interiorul unei singure încăperi. Adesea, această încăpere avea pereţi de sticlă prin care vizitatorii se puteau holba la marea minune electronică dinăuntru. O companie de mărime mijlocie sau o universitate ar fi putut avea unul sau două calculatoare, în timp ce instituţiile mari aveau cel mult câteva zeci. Ideea că, în mai puţin de 20 de ani, calculatoare la fel de puternice, mai mici decât un timbru poştal, vor fi produse pe scară largă în milioane de exemplare părea desprinsă dintr-un scenariu

ştiinţifico-fantastic.

Întrepătrunderea dintre domeniul calculatoarelor şi cel al comunicaţiilor a avut o influenţă profundă asupra modului în care sunt organizate sistemele de calcul. Conceptul de „centru de calcul” – în accepţiunea sa de încăpere unde există un calculator mare la care utilizatorii vin să-şi ruleze programele – este total depăşit. Vechiul model al unui singur calculator care serveşte rezolvării problemelor de calcul ale organizaţiei a fost înlocuit de un model în care munca este făcută de un număr mare de calculatoare separate, dar interconectate. Aceste sisteme se numesc **reţele de calculatoare**. Proiectarea şi organizarea acestor reţele reprezintă subiectul acestei cărţi.

Pe parcursul cărţii vom folosi termenul „reţea de calculatoare” pentru a desemna o colecţie de calculatoare autonome interconectate folosind o singură tehnologie. Se spune despre două calculatoare că sunt interconectate dacă sunt capabile să schimbe informaţie între ele. Conectarea nu se face neapărat printr-un cablu de cupru; pot fi folosite în acest scop fibra optică, radiaţii infraroşii, microunde sau sateliţi de comunicaţii. Reţelele pot fi de dimensiuni, tipuri şi forme diferite, aşa cum vom vedea ceva mai târziu. Deşi poate să pară straniu, nici Internet-ul şi nici World Wide Web-ul (reţea de întindere mondială) nu sunt reţele de calculatoare. Dacă parcurgeţi cartea până la sfârşit va fi clar şi de ce. Răspunsul simplist este următorul: Internet-ul nu este o singură reţea, ci o reţea de reţele, iar WWW este un sistem distribuit care funcţionează peste nivelul Internet-ului.

În literatura de specialitate, se face deseori confuzie între o reţea de calculatoare şi un **sistem distribuit**. Deosebirea esenţială este aceea că într-un sistem distribuit, o colecţie de calculatoare independente este percepută de utilizatorii ei ca un sistem coerent unic. De obicei, el are un model sau o unică paradigmă care îl reprezintă pentru utilizatori. Adesea, un modul software aflat pe nivelul superior al sistemului de operare (numit **middleware**) este responsabil pentru implementarea acestui model. Un bun exemplu de sistem distribuit arhicunoscut este chiar World Wide Web, în care totul ia în cele din urmă forma unui document (pagina Web).

Într-o reţea de calculatoare, coerenta, modelul şi programele sunt absente. Utilizatorii au în faţa lor maşini locale, fără nici o intenţie de a face aceste staţii să arate şi să se comporte într-adevăr ca un sistem unic coerent. Dacă însă maşinile se deosebesc prin structurile hardware sau chiar prin sistemul de operare, acest amănunt este vizibil pentru utilizatori. Dacă un utilizator doreşte să ruleze un program, el trebuie să se înregistreze pe maşina respectivă şi să lucreze acolo.

De fapt, un sistem distribuit este un sistem de programe construit peste o reţea. Programele asigură reţelei un grad mare de coeziune şi transparenţă. De aceea, diferenţa majoră între o reţea şi un sistem distribuit nu apare la nivel de echipamente, ci de programe (în special la nivelul sistemului de operare).

Nu mai puţin adevărat este faptul că între cele două subiecte există o suprapunere considerabilă. De exemplu, atât sistemele distribuite cât şi reţelele de calculatoare au nevoie să transfere fişiere. Diferenţa se referă la cine invocă transferul: sistemul sau utilizatorul. Deşi această carte are în vedere în primul rând reţelele, multe din subiectele abordate sunt importante şi în sistemele distribuite.

Pentru mai multe informaţii despre sistemele distribuite, a se vedea (Tanenbaum şi Van Steen, 2002).

1.1 UTILIZĂRILE REŢELELOR DE CALCULATOARE

Înainte de examinarea în detaliu a problemelor tehnice, merită să arătăm de ce sunt oamenii interesaţi de reţelele de calculatoare şi la ce pot fi ele folosite. Până la urmă, dacă nimeni nu ar fi interesat de reţele de calculatoare, puţine reţele ar fi construite. Vom începe cu utilizările tradiţionale în cadrul companiilor şi pentru utilizatorii individuali, apoi ne vom deplasa spre dezvoltările recente privind utilizatorii mobili şi reţelele domestice.

1.1.1 Aplicaţii comerciale

Multe companii au un număr semnificativ de calculatoare. De exemplu, o companie poate folosi calculatoare pentru monitorizarea producţiei, pentru urmărirea evoluţiei stocurilor, pentru calcularea statelor de plată. La început, fiecare din aceste calculatoare putea lucra izolat de celelalte, dar, la un moment dat, managerii au decis să le conecteze între ele pentru a putea extrage şi corela informaţii despre întreaga firmă.

În termeni mai generali, subiectul se referă la **împărţirea resurselor**, iar scopul este de a face toate programele, echipamentele şi în special datele disponibile pentru oricine din reţea, indiferent de localizarea fizică a resursei şi a utilizatorului. Un exemplu uzual şi larg răspândit este existenţa unui grup de utilizatori care folosesc o imprimantă comună. Nici unul dintre utilizatori nu are nevoie de propria imprimantă, iar o imprimantă performantă de volum mare, legată în reţea este, de cele mai multe ori, mai ieftină, mai rapidă şi mai uşor de întreţinut decât o colecţie de imprimante individuale.

Cu toate acestea, probabil chiar mai importantă decât partajarea resurselor fizice, cum sunt imprimantele, scannerele, dispozitivele de inscripţionat CD-uri, este partajarea informaţiei. Orice companie mare sau medie, dar şi multe dintre companiile mici sunt total dependente de informaţia prelucrată de calculatoare. Cele mai multe companii ţin înregistrările clienţilor, inventarele, evidenţa conturilor de încasări, rapoartele financiare, informaţiile despre taxe şi încă multe altele numai cu ajutorul calculatorului. Dacă toate calculatoarele sale se defectează, o bancă nu mai poate funcţiona mai mult de 5 minute. O fabrica modernă, cu o linie de asamblare condusă de calculator nu ar putea continua lucrul nici măcar atât. Chiar şi o mică agenţie de turism sau un birou de avocatură cu trei angajaţi sunt, în acest moment, dependente în mare măsură de reţelele de calculatoare, care le permit angajaţilor accesul instantaneu la informaţii relevante şi la documente.

Pentru companiile mai mici, toate calculatoarele sunt cel mai probabil amplasate într-un singur birou sau poate într-o singură clădire, în timp ce pentru companiile mai mari calculatoarele şi anga- jaţii pot fi răspândiţi într-o mulţime de birouri şi fabrici din diferite ţări. Cu toate acestea, un agent de vânzări din New York poate avea uneori nevoie de acces la o bază de date cu inventarul produselor aflată în Singapore. Cu alte cuvinte, numai faptul ca un utilizator se află la 15.000 km de datele de care are nevoie nu îl poate împiedica să-şi folosească datele ca şi când ele ar fi locale. Pe scurt, scopul poate fi definit ca o încercare de a termina cu „tirania geografiei".

În termenii cei mai simpli se poate imagina sistemul informaţional al unei companii ca fiind alcătuit din una sau mai multe baze de date şi un număr de angajaţi care au nevoie de acces de la distanţă. În acest model, datele sunt memorate în calculatoare performante, numite **servere (servers)**. Adesea, acestea sunt plasate şi întreţinute centralizat de un administrator de sistem. Din contră, angajaţii au maşini mai simple, numite **clienţi (clients)**, plasate pe birourile lor, prin intermediul cărora accesează datele aflate la distanţă pentru a le include, de exemplu, în foile de calcul pe care le construiesc. (Uneori ne vom referi la operatorul care foloseşte o maşină client cu numele de „client", dar va fi clar din context dacă referirea este la maşină sau la utilizatorul ei). Maşinile server şi client sunt conectate în reţea, aşa cum este ilustrat în fig. l-1. De notat că am reprezentat reţeaua ca un simplu oval, fără nici un alt detaliu. Vom mai folosi această formă pentru a reprezenta o reţea în mod abstract. Atunci când sunt necesare mai multe detalii, ele vor fi furnizate.

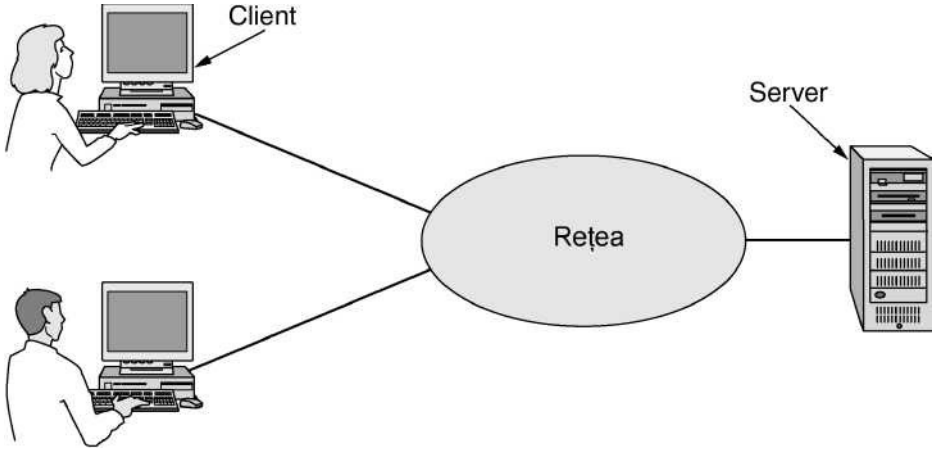


Fig. l-1. O reţea cu doi clienţi şi un server.

Această structură reprezintă **modelul client-server**. Este folosit frecvent şi reprezintă baza pe care lucrează multe reţele. Este aplicabil atunci când clientul şi serverul se află în aceeaşi clădire (de exemplu, dacă ambele aparţin aceleiaşi companii), dar şi atunci când între ele este o distanţă mai mare. De exemplu, atunci când o persoană aflată acasă face un acces la o pagină Web, este folosit acelaşi model, în care serverul Web aflat la distanţă are rol de server, iar calculatorul personal al utilizatorului are rol de client. În cele mai multe situaţii, un server poate lucra cu un număr mare de clienţi.

Dacă privim mai în detaliu modelul client-server, constatăm că sunt implicate două procese, unul aflat pe maşina client şi unul aflat pe maşina server. Comunicaţia ia forma transmiterii prin reţea a unui mesaj de la procesul client către procesul server. În continuare, procesul client va aştepta un mesaj de răspuns. Atunci când procesul server primeşte cererea, execut ă acţiunea solicitat ă sau caută datele cerute şi transmite un răspuns. Aceste mesaje sunt prezentate în fig. l-2.

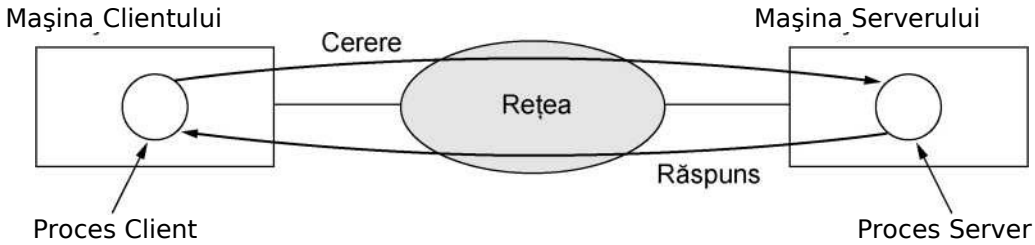


Fig. l-2. Modelul client-server implică cereri şi răspunsuri.

Un al doilea scop al construirii unei reţele de calculatoare este mai mult legat de oameni decât de informaţie sau chiar calculatoare. O reţea de calculatoare poate constitui un puternic **mediu de comunicare** între angajaţi. Aproape orice companie care are două sau mai multe calculatoare are acum **poştă electronică** (**e-mail**), pe care angajaţii o folosesc intens pentru comunicaţiile zilnice. De fapt, una dintre neplăcerile discutate intens între angajaţi este multitudinea de mesaje, în mare

parte lipsite de sens, cu care trebuie să se confrunte zilnic pentru că şefii au descoperit că pot trimite acelaşi mesaj (de cele mai multe ori chiar fără conţinut) tuturor subordonaţilor, prin apăsarea unui singur buton.

Dar poşta electronică nu este singura formă de comunicaţie îmbunătăţită care a fost făcută posibilă de reţelele de calculatoare. Cu o reţea, este uşor pentru doi oameni care lucrează la mare distanţă unul de altul să scrie un raport împreună. Când unul dintre ei face o modificare asupra unui document din reţea, ceilalţi vor putea vedea modificarea imediat, în loc să aştepte o scrisoare timp de mai multe zile. O astfel de accelerare face din cooperarea în cadrul grupurilor de oameni aflaţi la distanţă o simplă comunicare, fapt imposibil cu ceva timp în urmă.

O altă formă de comunicare asistată de calculator o reprezintă videoconferinţele. Folosind această tehnologie, angajaţii din locuri aflate la distanţă pot ţine o întrunire, pot să se vadă şi să se audă între ei, şi pot scrie chiar pe o tablă virtuală partajată. Videoconferinţa este o modalitate eficientă de eliminare a costurilor şi timpului pierdute anterior pentru a călători. Se spune uneori că între comunicare şi transport este o competiţie şi că activitatea care câştigă o face pe cealaltă să pară depăşită.

Un al treilea scop pentru tot mai multe companii este realizarea electronică a comerţului cu alte companii, în special cu furnizorii şi clienţii. De exemplu, producătorii de automobile, avioane sau calculatoare, printre alţii, cumpără subansamble de la diverşi furnizori şi apoi le asamblează. Folosind reţelele de calculatoare, producătorii pot plasa comenzile electronic, după cum este nevoie. Posibilitatea de a plasa comenzi în timp real (dacă este nevoie) reduce necesitatea stocurilor mari şi sporeşte eficienţa.

Un al patrulea scop care devine din ce în ce mai important este realizarea de tranzacţii cu consumatorii prin Internet. Companiile aeriene, librăriile şi magazinele de muzică au descoperit că multor consumatori le place comoditatea de a-şi face cumpărăturile de acasă. În consecinţă, multe companii oferă on-line cataloage cu bunurile şi serviciile disponibile şi chiar primesc comenzi on-line. Este de aşteptat ca acest sector să se dezvolte rapid în continuare. El este numit **comerţ electronic (ecommerce, electronic commerce)**.

1.1.2 Aplicaţii domestice

În 1977 Ken Olsen era preşedinte al Digital Equipment Corporation, care era pe vremea aceea a doua companie în lume în vânzarea de calculatoare (după IBM). Atunci când a fost întrebat de ce Digital nu se implică mai mult în piaţa calculatoarelor personale, el a răspuns: „Nu există nici un motiv ca fiecare individ să aibă un calculator acasă.” Istoria a arătat că răspunsul a fost greşit, iar Digital nu mai există. De ce cumpără oamenii calculatoare pentru a le folosi acasă? La început, pentru prelucrarea de texte şi pentru jocuri, dar în ultimii ani această imagine s-a schimbat radical. Probabil că în acest moment cel mai important motiv este accesul la Internet. Unele dintre cele mai populare utilizări ale Internet-ului pentru utilizatorii casnici sunt următoarele:

1. Accesul la informaţie de la distanţă.
2. Comunicaţiile interpersonale.
3. Divertismentul interactiv
4. Comerţul electronic

Accesul informaţiei la distanţă ia forme multiple. Poate fi navigarea pe Web pentru informaţii sau doar pentru distracţie. Categoriile de informaţii disponibile includ artele, afacerile, gastronomia, guvernarea, sănătatea, istoria, preocupările din timpul liber, modalităţile de recreere, ştiinţa, sporturile, călătoriile, şi multe altele. Distracţia este de prea multe feluri ca să poată fi menţionate, plus câteva care e mai bine să rămână nemenţionate.

Multe ziare sunt acum disponibile on-line şi pot fi personalizate. De exemplu, este uneori posibil să spui unui ziar că doreşti să obţii totul despre politicienii corupţi, despre marile incendii, despre scandalurile în care sunt implicate celebrităţile şi despre epidemii, dar nu despre fotbal. Uneori este chiar posibil să vă aduceţi articolele selectate pe discul local, în timp ce dormiţi, sau să le tipăriţi înainte de micul dejun. Şi cum această tendinţă continuă să se dezvolte, va cauza o creştere importantă a ratei şomajului printre băieţii de 12 ani care distribuie ziare, dar redacţiilor ziarelor le place această variantă, pentru că distribuţia a fost întotdeauna cea mai slabă verigă din întregul lanţ de producţie.

Pasul următor după ziare (împreună cu revistele şi jurnalele ştiinţifice) este biblioteca digitală online. Multe organizaţii profesionale, cum sunt ACM *(www.acm.org)* şi IEEE Computer Society *(www.computer.org)* au deja disponibile on-line multe dintre jurnale şi prezentări de la conferinţe. Alte grupuri urmează rapid această tendinţă. În funcţie de costul, dimensiunile şi greutatea unui calculator portabil, cărţile tipărite vor deveni desuete. Scepticii ar trebui să fie atenţi la efectul pe care l-a avut tiparul asupra manuscriselor medievale iluministe.

Toate aceste aplicaţii presupun interacţiuni între o persoană şi o bază de date aflată la distanţă. O a doua categorie largă de utilizări ale reţelei este comunicarea între persoane – este vorba în primul rând de replica secolului XXI la telefonul din secolul al XIX-lea. Poşta electronică, sau **e-mail**- ul, este deja folosită zi de zi de milioane de oameni din toată lumea şi gradul de utilizare este în continuă creştere. Conţine deja, în mod curent, pe lângă text şi poze, secvenţe audio şi video. În schimb, va dura ceva mai mult până când se va pune la punct înglobarea mirosului în mesaje.

Orice adolescent este dependent de **mesageria instantanee** (**instant messaging**). Această facilitate, derivată din programul UNIX *talk* (ro: vorbeşte) folosit încă din anii 1970, le permite celor doi care doresc să comunice să-şi trimită mesaje unul altuia în timp real. O versiune multipersonală a acestei idei este **chat-room-ul** (ro: **camera de discuţii**) în care o persoană dintr-un grup poate trimite mesaje către întregul grup.

Grupurile de ştiri de pe tot globul, cu discuţii privind orice subiect imaginabil, fac deja parte din realitatea cotidiană a unei anumite categorii de persoane, iar acest fenomen va creşte până la dimensiunile întregii omeniri. Discuţiile, în care o persoană trimite un mesaj şi toţi ceilalţi abonaţi ai grupului de interes pot să-l

citească, se derulează în toate stilurile posibile, putând fi la fel de bine extrem de amuzante sau de pătimaşe. Spre deosebire de camerele de discuţii (chatroom-uri), grupurile de interese nu sunt în timp real şi mesajele sunt salvate astfel încât atunci când cineva se întoarce din vacanţă, toate mesajele care au fost primite între timp aşteaptă cuminţi să fie citite.

Un alt tip de comunicaţie interpersonală se numeşte adesea comunicaţie **de la egal-la-egal** (**peer- to-peer**), pentru a o distinge de modelul client-server (Parameswaran et al., 2001). În această formă, persoanele independente care formează un grup oarecare comunică în cadrul grupului, după cum se vede în fig. l-3. Fiecare persoana poate, în principiu, să comunice cu una sau mai multe persoane; nu există o departajare clară între clienţi şi servere.

Comunicaţiile de la egal-la-egal au explodat în jurul anului 2000 cu un serviciu numit Napster, care la apogeu avea peste 50 de milioane de fani ai muzicii care schimbau între ei melodii. A fost probabil cea mai mare înfrângere a drepturilor de autor din toată istoria lor (Lam şi Tan, 2001; şi Macedonia, 2000). Ideea era destul de simplă. Membrii înregistrau muzica pe care o aveau pe discurile locale într-o bază de date centrală întreţinută de serverul Napster. Dacă un membru dorea o melodie, verifica baza de date ca să vadă cine o are şi se ducea direct la sursă pentru a o lua. Şi pentru că Napster nu ţinea nici un fel de muzică pe maşinile proprii, Napster a argumentat că nu a încălcat drepturile de autor ale nimănui. Dar tribunalul nu a fost de acord şi a închis sistemul.

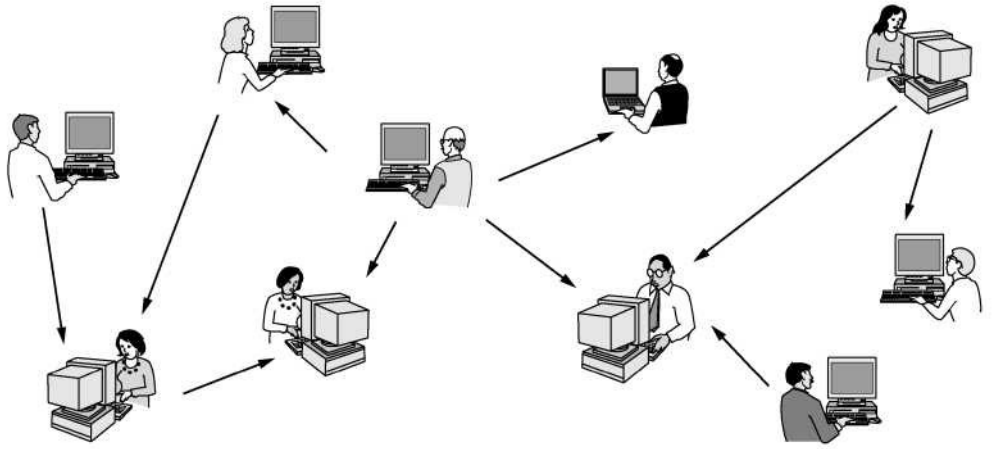


Fig. l-3. Într-un sistem de la egal la egal nu sunt clienţi şi servere fixe.

Oricum, următoarea generaţie de sisteme de la egal-la-egal elimină baza de date centrală deoarece fiecare utilizator îşi va întreţine propria bază locală şi va oferi o listă de alţi utilizatori membri ai sistemului aflaţi în apropiere. Un nou utilizator va putea atunci să viziteze fiecare membru şi să vadă ce anume are acesta şi care este lista de utilizatori aflaţi în apropierea sa. Acest proces de căutare poate fi repetat la infinit pentru a crea o bază de date de dimensiune mare cu ceea ce se regăseşte în sistem. Este o activitate care ar deveni tracasantă pentru utilizatori, dar pentru care calculatoarele sunt excelente.

Există de asemenea şi aplicaţii legale pentru comunicaţiile de la egal-la-egal. De exemplu, fanii partajează muzica neprotejată de drepturile de autor sau noile extrase de melodii pe care formaţiile muzicale le oferă în scop publicitar, familiile partajează poze, filme şi informaţii genealogice, iar adolescenţii joacă on-line jocuri cu mai mulţi participanţi. De fapt, una dintre cele mai populare aplicaţii ale Internet-ului, poşta electronică, este în mod implicit de la egal-la-egal. Este de aşteptat ca această formă de comunicaţie să crească semnificativ în viitor.

Criminalitatea electronică nu este limitată la încălcarea drepturilor de autor. O altă zonă fierbinte este cea a jocurilor de noroc electronice. Calculatoarele au simulat tot felul de lucruri timp de decenii. De ce să nu simuleze şi automatele cu fise, roata ruletei, masa de blackjack, şi multe alte echipamente pentru jocurile de noroc? Ei bine, deoarece este ilegal în multe locuri. Problema este că jocurile de noroc sunt legale în multe alte părţi (în Anglia, de exemplu) şi proprietarii de cazinouri din astfel de state au înţeles potenţialul jocurilor de noroc pe Internet. Ce se întâmplă dacă jucătorul şi cazinoul se află în ţări diferite, cu legi diferite? Bună întrebare.

Alte aplicaţii orientate pe comunicaţii includ utilizarea Internet-ului ca suport pentru convorbiri telefonice, conferinţe video sau radio, trei domenii în plină dezvoltare. O altă aplicaţie este învăţământul la distanţă, aceasta însemnând ca poţi să urmăreşti cursurile de la 8 dimineaţa fără a trebui să te dai mai întâi jos din pat. Pe termen lung, utilizarea calculatoarelor pentru a îmbunătăţi comunicaţiile interumane se va putea dovedi mai importantă decât oricare alte utilizări.

A treia categorie avută în vedere este divertismentul, care reprezintă o industrie uriaşă, în continuă creştere. În acest domeniu aplicaţia de cel mai mare succes (cea care poate să influenţeze tot restul) se numeşte video la cerere. Este plauzibil ca peste vreo zece ani să putem selecta orice film sau program de televiziune realizat vreodată în orice ţară şi acesta să fie imediat disponibil pe ecranul nostru. Filmele noi ar putea deveni interactive: spectatorul ar fi întrebat în anumite momente ce continuare a povestirii alege (să-l ucidă MacBeth pe Duncan sau să aştepte o ocazie mai bună?), fiind prevăzute scenarii alternative pentru toate cazurile. De asemenea, televiziunea în direct s-ar putea desfăşura interactiv, cu telespectatori care participă la concursuri, care aleg câştigătorul dintre concurenţii preferaţi şi aşa mai departe.

Pe de altă parte, poate că nu sistemul de video la cerere, ci jocurile vor reprezenta aplicaţia de maxim succes. Există deja jocuri pentru mai multe persoane cu simulare în timp real, de exemplu v-aţi ascuns într-o închisoare virtuală sau simulatoare de zbor în care jucătorii unei echipe încearcă să-i doboare pe cei din echipa adversă. Dacă jocurile sunt jucate cu ochelari pentru realitatea virtuală, în medii tridimensionale, în timp real şi cu imagini de calitate fotografică, atunci avem un fel de realitate virtuală globală şi partajată.

Cea de-a patra categorie este comerţul electronic în cel mai larg sens al cuvântului. Cumpărăturile făcute de acasă sunt deja populare şi permit utilizatorilor să inspecteze on-line cataloagele a mii de companii. Unele dintre aceste cataloage vor oferi în curând posibilitatea de a obţine o prezentare video imediată a oricărui produs printr-o simplă selectare a numelui produsului. După ce

un client cumpără electronic un produs, dar nu poate să îşi dea seama cum să îl folosească, poate fi consultat departamentul de ajutor on-line.

O altă arie de interes în care comerţul electronic este deja implementat este accesul la instituţiile financiare. Mulţi oameni îşi plătesc facturile, îşi administrează conturile bancare şi îşi manevrează investiţiile electronic. Acestea se vor dezvolta şi mai repede de îndată ce reţelele vor deveni mai sigure.

O zonă de interes pe care nimeni nu o întrevedea ca interesantă este talciocul electronic (flea market). Licitaţiile on-line de bunuri la mâna a doua au devenit o industrie uriaşă. Spre deosebire de comerţul electronic tradiţional, care este construit după modelul client-server, licitaţiile on-line sunt mai aproape de sistemul de la egal-la-egal, un fel de consumator-la-consumator. Unele dintre aceste forme de comerţ electronic au obţinut porecle simpatice, plecând de la faptul că, în limba engleză, „2 (two) ” şi „to” se pronunţă la fel. Cele mai populare sunt prezentate în fig. l-4.

Prescurtar	Nume întreg	Exemplu
B2C	Companie la Consumator (Bussiness to Consumer)	Comanda de cărţi online
B2B	Companie la Companie (Bussiness to Bussiness)	Fabricantul de maşini comandă cauciucuri de la furnizor
G2C	Guvern la consumator (Government to Consumer)	Guvernul distribuie formularele pentru taxe în format electronic
C2C	Consumator la Consumator (Consumer to Consumer)	Licitarea de produse mâna a doua online
P2P	Punct la Punct (Peer-to-Peer)	Partajare de fişiere

Fig. l-4. Unele forme de comerţ electronic.

Fără îndoială că domeniile de utilizare pentru reţelele de calculatoare se vor dezvolta încă şi mai mult în viitor şi probabil că vor aborda direcţii pe care acum nu le poate prevedea nimeni. La urma urmei, câţi oameni ar fi crezut în 1990 că adolescenţii care îşi scriu plictisiţi mesaje pe telefoanele mobile în timp ce călătoresc cu autobuzul vor deveni o imensă sursă de bani pentru companiile de telefonie mobilă? Cu toate acestea, serviciul de mesaje scurte este extrem de profitabil.

Reţelele de calculatoare pot deveni foarte importante pentru oamenii care se află în locuri mai greu accesibile cărora le pot oferi accesul la aceleaşi servicii la disponibile şi celor care stau în centrul oraşelor. Învăţământul la distanţă poate afecta hotărâtor educaţia; universităţile vor deveni naţionale sau chiar internaţionale. Medicina la distanţă este abia la început (de exemplu monitorizarea pa- cienţilor de la distanţă), dar poate să devină mult mai importantă. Dar aplicaţia cea mai de succes poate să fie ceva mai practică, cum ar fi folosirea unei camere digitale în frigider pentru a vedea dacă trebuie să cumperi lapte când vii acasă de la serviciu.

1.1.3 Utilizatorii mobili

Calculatoarele mobile, cum sunt portabilele sau PDA-urile (**Personal Digital Assistant**, rom: asistent digital personal) sunt unele dintre segmentele cu

dezvoltarea cea mai rapidă din industria calculatoarelor. Mulţi posesori ai acestor calculatoare au calculatoare la birou şi doresc să fie conectaţi la ele chiar şi când sunt plecaţi de acasă sau pe drum. Şi cum a avea o conexiune pe fir este imposibil în maşini sau în avioane, există un interes deosebit pentru reţelele fără fir. În această secţiune vom studia pe scurt câteva dintre aplicaţiile reţelelor fără fir.

De ce şi-ar dori cineva o astfel de reţea? Unul dintre motivele uzuale este că obţine un birou portabil. Oamenii care călătoresc mult doresc să-şi poată folosi echipamentele electronice portabile pentru a trimite şi pentru a primi apeluri telefonice, faxuri şi poşta electronică, pentru a naviga pe Web, pentru a accesa fişiere la distanţă şi pentru a se putea conecta la maşini aflate la distanţă. Şi vor să poată face toate acestea în orice loc de pe Pământ, de pe mare sau din aer. De exemplu, în ultima vreme, la conferinţele legate de calculatoare organizatorii setează o reţea locală fără fir în încăperea în care se ţin conferinţele. Oricine are un calculator portabil cu un modem fără fir va trebui doar să îşi pornească propriul calculator pentru a fi conectat la Internet, ca şi cum calculatorul ar fi conectat cu un fir într-o reţea obişnuită. Similar, unele universităţi au instalat reţele fără fir în campus, astfel încât studenţii să poată sta la umbra copacilor şi să consulte catalogul bibliotecii sau să-şi citească poşta electronică.

Reţelele fără fir sunt de mare valoare pentru parcurile de taximetre, camioane, vehicule utilizate pentru livrare şi chiar echipe de intervenţie, pentru a fi mereu în contact cu baza. De exemplu, în multe oraşe şoferii de taxi sunt oameni de afaceri independenţi, nu angajaţi ai unei companii de taximetre. În unele dintre aceste oraşe, taximetrele au un ecran pe care şoferul îl poate vedea. Când sună un client, un dispecer central introduce locul de unde trebuie preluat clientul şi destinaţia unde acesta doreşte să ajungă. Această informaţie este afişată pe ecranul din taximetru şi se generează un semnal sonor. Primul şofer care atinge un buton al ecranului este cel care preia apelul.

Reţelele fără fir sunt de asemenea importante în domeniul militar. Dacă vrei să porneşti un război oriunde în lume într-un termen scurt, a conta pe infrastructura de reţea de la faţa locului nu este, cel mai probabil, o idee bună. Este mai bine să o aduci pe a ta de acasă.

Deşi reţelele fără fir şi calculatoarele mobile sunt deseori în strânsă legătură, ele nu sunt domenii identice, după cum arată şi fig. l-5. Aici se vede diferenţa între **fix fără fir** şi **mobil fără fir**. Chiar şi calculatoarele portabile au uneori nevoie de cablu. De exemplu, dacă un călător conectează firul de la calculatorul său portabil în priza de telefon din camera de hotel, el are mobilitate, folosindu-se totuşi de cablu.

Fără fir	Mobil	Aplicaţii
Nu	Nu	Cal culatoarele staţionare de pe mesele de lucru din birouri
Nu	Da	Un calculator portabil folosit într-o camera de hotel
Da	Nu	Reţelele în clădiri mai vechi, necablate
Da	Da	Biroul portabil; PDA pentru inventarul magazinului

Fig. l-5. Combinaţii de reţele fără fir şi echipamente mobile.

Pe de altă parte, unele calculatoare fără fir nu sunt mobile. Un exemplu important este o companie care are o clădire mai veche, necablată pentru reţea şi doreşte să îşi interconecteze calculatoarele. Instalarea unei reţele fără fir necesită

doar puţin mai mult decât a cumpăra o cutie care are ceva electronică, a o despacheta şi a o conecta. Totuşi, această soluţie poate fi mult mai ieftină decât a pune un tehnician să tragă cabluri pentru a cabla întreaga clădire.

Există, desigur, aplicaţii cu adevărat mobile şi fără fire, de la birourile portabile până la oamenii care, intrând în magazin cu un PDA pot face inventarul. La multe aeroporturi aglomerate, oamenii care se ocupă de primirea maşinilor care au fost închiriate lucrează cu ajutorul calculatoarelor portabile fără fire. Ei introduc numărul de înmatriculare al maşinilor care sunt returnate şi echipamentul portabil, care are o imprimantă ataşată, apelează calculatorul principal, obţine informaţiile despre închiriere şi tipăreşte pe loc factura.

Pe măsură ce tehnologiile de comunicaţie fără fir devin din ce în ce mai răspândite, sunt pe cale să apară tot mai multe aplicaţii. Să analizăm rapid unele posibilităţi. Aparatele de taxat fără fir pentru plata parcării au avantaje atât pentru utilizatori cât şi pentru mai marii oraşului. Aparatele de taxat pot să accepte cărţi de credit sau de debit şi să le verifice imediat prin conexiunea fără fir. Când perioada pentru care s-a plătit expiră, aparatul poate să verifice existenţa unei maşini în locul de parcare (va trimite un semnal înspre ea şi, dacă acesta este reflectat, în spaţiul respectiv se găseşte o maşină) şi să raporteze poliţiei eventuala depăşire. S-a estimat că, numai la nivelul oraşelor din SUA, municipalităţile ar putea obţine un plus de 10 miliarde de dolari folosind această variantă (Harte et al., 2000). Mai mult, sancţionarea mai riguroasă pentru parcarea ilegală va ajuta mediul înconjurător, deoarece şoferii care ştiu că vor fi prinşi în cazul în care parchează ilegal ar putea să folosească transportul în comun.

Automatele de gustări, băuturi şi alte bunuri se găsesc peste tot. Desigur, mâncarea nu ajunge în aceste automate prin puterea magiei. Periodic, cineva vine cu un camion pentru a le umple. Dacă automatele însele ar transmite printr-o conexiune fără fir un raport în fiecare zi pentru a comunica stocurile curente, şoferul camionului ar şti ce maşini trebuie re-aprovizionate şi ce cantitate din fiecare produs trebuie să aducă. O astfel de informaţie ar duce la o planificare mai eficientă a drumului. Desigur, această informaţie ar putea să fie transmisă şi prin liniile telefonice standard, dar soluţia de a da fiecărui automat o conexiune fixă de telefon pentru un singur apel pe zi este scumpă din cauza taxei lunare fixe.

O altă zonă în care tehnologiile de conectare fără fir pot să ducă la economii sunt citirile contoarelor pentru diverse utilităţi. Varianta în care consumul la energie electrică, gaze, apă, şi alte utilităţi care se regăsesc în casele oamenilor ar putea să fie raportat folosind o astfel de conexiune fără fir, nu ar mai fi nevoie să fie trimişi pe teren angajaţi care să se ocupe de citirea contoarelor. Similar, detectoarele de fum fără fir ar putea să sune la divizia de Pompieri în loc să facă un zgomot infernal (care este lipsit de orice valoare dacă nu este nimeni acasă). Deoarece costul dispozitivelor radio şi cel al timpului de emisie scad, din ce în ce mai multe măsurători se vor face prin intermediul reţelelor fără fire.

O arie de aplicaţii complet diferită pentru reţelele fără fir este mult aşteptata fuziune între telefoanele mobile şi PDA-uri în mici calculatoare fără cablu. O primă încercare a fost făcută cu micile PDA-uri, care puteau să afişeze pagini Web simplificate pe minusculele lor ecrane. Acest sistem, numit WAP 1.0 (Wireless

Application Protocol, rom: protocolul aplicaţiilor fără fir) a eşuat, tocmai din cauza ecranelor prea mici, a lărgimii de bandă scăzute şi a serviciilor slabe calitativ. Dar dispozitivele şi serviciile mai noi vor funcţiona mai bine cu WAP 2.0.

O zonă în care aceste dispozitive pot fi excelente este denumită comerţ mobil **(m-commerce)** (Senn, 2000). Forţa care stă în spatele acestui fenomen constă dintr-un amalgam de producători de dispozitive PDA fără fir şi operatori de reţea care încearcă din răsputeri să găsească o soluţie pentru a obţine o bucată din plăcinta comerţului electronic. Una dintre speranţele lor este să folosească PDA-urile fără fir pentru operaţiuni bancare şi pentru cumpărături. O idee este utilizarea PDA- urilor ca pe un fel de portofel electronic, autorizând plăţile în magazine, ca un înlocuitor pentru banii lichizi şi pentru cărţile de credit. Suma cheltuită apare apoi pe factura telefonului mobil. Din punct de vedere al magazinelor, această schemă aduce un câştig prin economisirea taxelor plătite companiei de cărţi de credit, taxă care poate fi de câteva procente. Desigur, acest plan poate fi dezavantajos, deoarece clienţii dintr-un magazin îşi pot folosi PDA-urile pentru a verifica preţurile concurenţei înainte de a cumpăra. Încă şi mai rău, companiile de telefoane pot oferi PDA-uri cu cititoare de coduri de bare care să permită unui client să scaneze un produs dintr-un magazin şi apoi să obţină instantaneu un raport detaliat despre alte locuri în care acelaşi produs se găseşte şi despre preţul lui.

Deoarece operatorul reţelei ştie unde anume se găseşte utilizatorul, unele servicii sunt în mod intenţionat dependente de loc. De exemplu, poate fi posibil să afli localizarea unui magazin de cărţi sau a unui restaurant chinezesc din apropiere. Hărţile mobile sunt un alt candidat. La fel sunt şi prognozele meteo foarte localizate („Când o să se oprească ploaia în curtea mea din spate?”). Fără îndoială că multe alte aplicaţii or să apară pe măsură ce aceste dispozitive devin tot mai răspândite.

Unul dintre lucrurile importante după care comerţul mobil s-a orientat este acela că utilizatorii de telefoane mobile sunt obişnuiţi să plătească pentru tot (spre deosebire de utilizatorii de Internet, care aşteaptă totul gratis). Dacă un sit Internet ar impune o taxă pentru a permite utilizatorilor săi să plătească prin intermediul cărţii de credit, s-ar naşte o grămadă de proteste zgomotoase din partea utilizatorilor. Dacă un operator de telefonie mobilă ar permite oamenilor să plătească pentru articolele dintr-un magazin folosind telefonul şi apoi le-ar fi impus o taxă pentru acest serviciu, probabil că totul ar fi fost perceput ca normal. Timpul va decide.

Ceva mai departe în timp sunt reţelele personale (personal area networks) şi calculatoarele la purtător (wearable computers). IBM a dezvoltat un ceas care rulează Linux (inclusiv sistemul de ferestre X11) şi are conexiune fără fir la Internet pentru a trimite şi primi mesaje prin poşta electronică (Narayanaswami et al., 2002). În viitor, oamenii vor putea schimba cărţi de vizită numai prin punerea ceasurilor lor faţă în faţă. Calculatoarele la purtător, fără fir, vor putea permite accesul oamenilor în încăperi securizate în acelaşi fel în care cardurile cu benzi magnetice o fac astăzi (probabil că vor lucra în combinaţie cu un cod PIN sau cu măsurători biometrice). Este posibil ca aceste ceasuri să fie capabile chiar să obţină informaţiile relevante în vecinătatea utilizatorului (de exemplu restaurante locale). Posibilităţile sunt infinite.

Ceasurile inteligente cu radio au fost parte din spaţiul nostru mental încă de când au apărut în benzile comice cu Dick Tracy în 1946. Dar praful inteligent? Cercetătorii de la Berkley au construit un calculator fără fir într-un cub cu latura de 1 mm (Warneke et al., 2001). Aplicaţiile potenţiale includ evidenţa stocurilor, pachetelor, ba chiar şi a păsărelelor, rozătoarelor şi insectelor.

1.1.4 Aspecte sociale

Introducerea pe scară largă a reţelelor va ridica noi probleme sociale, etice şi politice. Vom menţiona pe scurt câteva dintre ele; un studiu exhaustiv ar necesita cel puţin o carte. O aplicaţie populară a multor reţele sunt grupurile de interese sau sistemele de informare în reţea (BBS-urile), unde oamenii pot schimba mesaje cu persoane având preocupări similare. Atâta vreme cât este vorba de subiecte tehnice sau de pasiuni precum grădinăritul, nu sunt motive să apară multe probleme.

Problemele se ivesc în cazul grupurilor de interese care iau în discuţie subiecte delicate sau extrem de disputate, cum ar fi politica, religia sau sexul. Atitudinile exprimate în cadrul acestor grupuri pot fi considerate ofensatoare de către anumiţi oameni. Mai mult chiar, nu este obligatoriu ca mesajele să se limiteze la text. Fotografii color de înaltă rezoluţie şi chiar scurte clipuri video pot fi acum transmise cu uşurinţă prin reţelele de calculatoare. Unii oameni au o atitudine neutră („trăieşte şi lasă-mă să trăiesc”), dar alţii consideră că trimiterea anumitor materiale (de exemplu, atacuri la anumite ţări sau religii, pornografia etc.) este pur şi simplu inacceptabilă şi trebuie cenzurată. Diverse ţări au diverse legi în acest domeniu, uneori chiar contradictorii. De aceea, discuţiile sunt în continuare aprinse.

Unii oameni au dat în judecată operatori de reţea, pretinzând că ei sunt responsabili pentru informaţia care circulă, exact ca în cazul ziarelor şi revistelor. Răspunsul inevitabil este că reţeaua e ca o companie de telefoane sau ca un oficiu poştal şi nu poate controla ceea ce discută utilizatorii săi. Mai mult chiar, dacă operatorii reţelei ar cenzura mesajele, atunci probabil că ei ar putea şterge orice fără a exista nici cea mai mică posibilitate de a-i da în judecată, încălcând astfel dreptul utilizatorilor la exprimare liberă. Nu este, probabil, hazardat să afirmăm că această dezbatere va continua mult timp.

O altă dispută animată are în atenţie drepturile angajaţilor în raport cu drepturile patronilor. Multe persoane citesc şi scriu poştă electronică la serviciu. Directorii unor firme au pretins că ar avea dreptul să citească şi eventual să cenzureze mesajele angajaţilor, inclusiv mesajele trimise de la calculatoarele de acasă, după orele de program. Numai că nu toţi angajaţii agreează această idee.

Dar chiar admiţând că directorii au o astfel de putere asupra angajaţilor, există o relaţie similară şi între universităţi şi studenţi? Dar între licee şi elevi? În 1994 Universitatea Carnegie-Mellon a hotărât să blocheze mesajele care veneau de la grupuri de interese legate de sex pe motivul că materialele nu erau potrivite pentru minori (adică pentru cei câţiva studenţi care nu aveau încă 18 ani). Disputa izvorâtă din această decizie va dura ani întregi.

Un alt subiect cheie este relaţia guvern-cetăţean. FBI a instalat la mulţi furnizori

de servicii Internet un sistem care să supravegheze toate mesajele de poştă electronică care vin şi pleacă în căutarea de amănunte din domeniile sale de interes (Blaze şi Bellovin, 2000; Sobel, 2001 şi Zacks, 2001). Sistemul a fost numit la început „Carnivore”, dar din cauza publicităţii negative de care a avut parte a fost redenumit cu un nume care suna ceva mai inocent: DCS1000. Dar scopul lui a rămas acelaşi: de a spiona milioane de oameni în speranţa că se vor găsi informaţii despre activităţi ilegale. Din păcate, al patrulea amendament al Constituţiei SUA interzice cercetările guvernamentale fără mandat de căutare. Dacă aceste 54 de cuvinte scrise în secolul al 18-lea au în continuare o oarecare valoare în secolul 21, tribunalele vor rămâne ocupate până în secolul 22.

Guvernul nu are monopol la ameninţarea intimităţii cetăţeanului. Sectorul privat îşi are şi el partea lui. De exemplu, micile fişiere denumite **cookies** (**prăjiturele**) pe care programele de navigare le stochează pe calculatoarele utilizatorilor permit companiilor să urmărească activităţile utilizatorilor în cyberspace şi, de asemenea, pot face ca numerele cărţilor de credit, numerele de asigurări sociale sau alte informaţii strict confidenţiale să fie accesibile în Internet (Berghel, 2001).

Reţelele de calculatoare oferă posibilitatea de a trimite mesaje anonime. În anumite situaţii aşa ceva este de dorit. De exemplu, reprezintă un mijloc pentru studenţi, soldaţi, angajaţi, cetăţeni de a trage un semnal de alarmă – fără teamă de represalii – în cazul comportamentului ilegal al profesorilor, ofiţerilor, directorilor sau politicienilor. Pe de altă parte, în Statele Unite şi în majoritatea democraţiilor, legea asigură în mod explicit dreptul unei persoane acuzate de a-şi chema acuzatorul în faţa Curţii. Acuzaţiile anonime nu pot servi drept probă.

Pe scurt, reţelele de calculatoare, asemenea industriei tipografice cu 500 de ani în urmă, permit cetăţenilor obişnuiţi să-şi lanseze opiniile prin mijloace diferite şi către audienţe diferite faţă de cele de până acum. Această libertate nou descoperită aduce cu ea probleme nerezolvate de ordin social, politic şi moral.

Odată cu binele vine şi răul. Viaţa pare a fi construită astfel. Internetul oferă posibilitatea de a găsi repede informaţii, dar multe dintre ele sunt greşit informate, tendenţioase sau chiar complet eronate. Sfatul medical pe care tocmai l-aţi luat de pe Internet poate să vină de la un laureat al premiului Nobel sau de la un repetent din liceu. Reţelele de calculatoare au introdus de asemenea şi noi tipuri de comportamente antisociale şi infracţionale. Transmiterea electronică a fleacurilor şi gunoaielor (eng.: junk) a devenit parte din viaţă pentru că oamenii au colecţionat milioane de adrese pe care le vând pe CD-ROM-uri aşa-zişilor agenţi de marketing. Mesajele care au un conţinut activ (de obicei programe sau macrouri care se execută pe maşina receptorului) pot avea efecte distructive.

Furtul de identitate devine o problemă serioasă, pentru că hoţii colectează destule informaţii despre o potenţială victimă pentru a putea obţine cărţi de credit şi alte documente în numele acesteia. În fine, posibilitatea de a transmite digital muzică şi filme a deschis uşa pentru încălcarea masivă a drepturilor de autor care sunt greu de depistat şi pedepsit.

Multe dintre aceste probleme puteau fi rezolvate dacă industria de calculatoare ar fi luat în serios securitatea calculatoarelor. Dacă toate mesajele erau criptate şi autentificate, ar fi fost mai greu să se comită nedreptăţi sau furturi. Această

tehnologie este bine conturată şi o vom studia în detaliu în cap. 8. Problema este că vânzătorii de hardware şi aplicaţii software ştiu că introducerea unor atribute de securitate costă bani, iar cumpărătorii nu solicită astfel de atribute. Mai mult, un număr substanţial de probleme este determinat de aplicaţiile care funcţionează cu erori, ceea ce se întâmplă pentru că producătorii adaugă din ce în ce mai multe facilităţi programelor lor, ceea ce înseamnă inevitabil mai mult cod şi de aceea mai multe erori. O taxă pentru noile facilităţi ar putea ajuta, dar ar face produsele greu de vândut în anumite segmente de piaţă. Plata unei despăgubiri pentru programele care funcţionează eronat ar fi foarte cinstită, doar că ar duce la faliment întreaga industrie software chiar din primul an.

1.2 HARDWARE-UL REŢELEI

A venit acum timpul să ne îndreptăm atenţia de la aplicaţiile şi problemele sociale ale interconectării (partea distractivă) la aspectele tehnice care intervin în proiectarea reţelelor (partea serioasă de lucru). Deşi nu există o taxonomie general acceptată în care pot fi încadrate toate reţelele de calculatoare, sunt extrem de importante două criterii: tehnologia de transmisie şi scara la care operează reţeaua. Vom examina pe rând fiecare din aceste aspecte.

În principal există două tipuri de tehnologii de transmisie care se folosesc pe scară largă. Acestea sunt:

1. Legături cu difuzare.
2. Legături punct-la-punct.

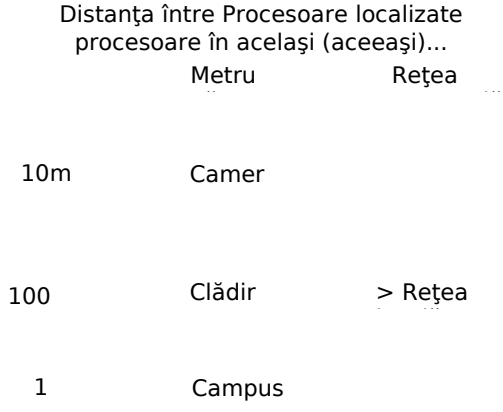
Reţelele cu difuzare au un singur canal de comunicaţii care este partajat de toate maşinile din reţea. Orice maşină poate trimite mesaje scurte, numite în anumite contexte **pachete**, care sunt primite de toate celelalte maşini. Un câmp de adresă din pachet specifică maşina căreia îi este adresat pachetul. La recepţionarea unui pachet, o maşină controlează câmpul de adres ă. Dacă pachetul îi este adresat, maşina îl prelucrează; dacă este trimis pentru o altă maşină, pachetul este ignorat.

Să considerăm, ca analogie, că cineva se află la capătul unui coridor cu multe încăperi şi strigă ,,Watson, vino aici: Am nevoie de tine.” Deşi pachetul poate fi primit (auzit) de multă lume, numai Watson va răspunde. Ceilalţi pur şi simplu îl ignoră. Un alt exemplu ar fi un aeroport unde se anunţă că toţi pasagerii zborului 644 sunt rugaţi să se prezinte la poarta 12.

Sistemele cu difuzare permit în general şi adresarea unui pachet către *toate* destinaţiile, prin folosirea unui cod special în câmpul de adresă. Un pachet transmis cu acest cod este primit şi prelucrat de toate maşinile din reţea. Acest mod de operare se numeşte **difuzare**. Unele sisteme cu difuzare suportă de asemenea transmisia la un subset de maşini, operaţie cunoscută sub numele de **trimitere multiplă**. Una din schemele posibile este să se rezerve un bit pentru a indica trimiterea multiplă. Restul de n – 1 biţi de adresă pot forma un număr de grup. O maşină se poate „abona” la orice grup sau la toate grupurile. Un pachet trimis unui anumit grup va ajunge la toate maşinile abonate la grupul respectiv.

Prin contrast, **reţelele punct-la-punct** dispun de numeroase conexiuni între perechi de maşini individuale. Pentru a ajunge de la sursă la destinaţie pe o reţea de acest tip, un pachet s-ar putea să fie nevoit să treacă prin una sau mai multe maşini intermediare. Deseori sunt posibile trasee multiple, de diferite lungimi, şi de aceea descoperirea drumurilor celor mai potrivite este foarte importantă. Ca o regulă generală (deşi există numeroase excepţii), reţelele mai mici, localizate geografic, tind să utilizeze difuzarea, în timp ce reţelele mai mari sunt de obicei punct-la-punct. Transmisiile punct la punct cu un sigur transmiţător şi un singur receptor sunt numite uneori şi **unicasting**.

Exempl



10	Oraş	Reţea
100	Ţară	Reţea larg răspândită
1000	Continen	
10.000	Planet	Internet-

Fig. l-6. Clasificarea procesoarelor interconectate în funcţie de dimensiune.

Un criteriu alternativ pentru clasificarea reţelelor este mărimea lor. În fig. l-6 este prezentată o clasificare a sistemelor cu procesoare multiple după mărimea lor fizică. Prima categorie o reprezintă reţelele personale (personal area networks), reţele gândite pentru o singură persoană. De exemplu, o reţea fără fir care conectează calculatorul cu perifericele sale (tastatură, imprimantă, mouse) este o reţea personală. De asemenea, un PDA care controlează aparatul auditiv al utilizatorului sau regulatorul lui de ritm cardiac se încadrează în aceeaşi categorie. Mai departe de aceste reţele personale sunt reţele cu domenii mai mari. Acestea pot fi împărţite în reţele locale, reţele metropolitane şi reţele larg răspândite geografic. În sfârşit, prin conectarea a două sau mai multe reţele rezultă o inter-reţea. Internet-ul este un exemplu bine cunoscut de inter-reţea. Distanţa este un criteriu de clasificare important, pentru că, la scări diferite, sunt folosite tehnici diferite. În această carte ne vom ocupa de reţele din toate aceste categorii. Prezentăm mai jos o scurtă introduce în subiectul echipamentelor de reţea.

1.2.1 Reţele locale

Reţelele locale (**Local Area Networks**)**,** denumite în general LAN-uri, sunt reţele private localizate într-o singură clădire sau într-un campus de cel mult câţiva kilometri. Ele sunt frecvent utilizate pentru a conecta calculatoarele personale şi staţiile de lucru din birourile companiilor şi fabricilor, în scopul de a partaja resurse (imprimante, de exemplu) şi de a schimba informaţii. LAN-urile se disting de alte tipuri de reţele prin trei caracteristici: (1) mărime, (2) tehnologie de transmisie şi (3) topologie.

LAN-urile au dimensiuni restrânse, ceea ce înseamnă că timpul de transmisie în cazul cel mai defavorabil este limitat şi cunoscut dinainte. Cunoscând această limită, este posibil să utilizăm anumite tehnici de proiectare care altfel nu ar fi fost posibile. Totodată, se simplifică administrarea reţelei.

LAN-urile utilizează frecvent o tehnologie de transmisie care constă dintr-un singur cablu la care sunt ataşate toate maşinile, aşa cum erau odată cablurile telefonice comune în zonele rurale. LAN-urile tradiţionale funcţionează la viteze cuprinse între 10 şi 100 Mbps, au întârzieri mici (mi- crosecunde sau nanosecunde) şi produc erori foarte puţine. LAN-urile mai noi pot opera la viteze mai mari, până la 10 Gbps. În această carte vom păstra tradiţia şi vom măsura vitezele de transmi-sie pe linii în megabiţi/sec (1 Mbps reprezintă 1.000.000 biţi), şi gigabiţi/sec (1 Gbps reprezintă 1.000.000.000 biţi). ’ ’ ’

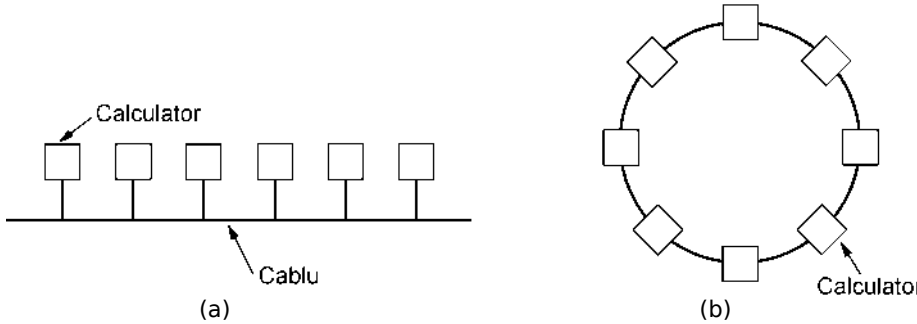


Fig. l-7. Două reţele cu difuzare. (a) Magistrală. (b) Inel.

Pentru LAN-urile cu difuzare sunt posibile diverse topologii. Fig. l-7 prezintă două dintre ele. Într-o reţea cu magistrală (cu cablu liniar), în fiecare moment cel mult una dintre maşini este master şi are dreptul să transmită. Restul maşinilor nu pot transmite. Când două sau mai multe maşini vor să transmită simultan, este necesar un mecanism de arbitrare. Mecanismul de arbitrare poate fi centralizat sau distribuit. De exemplu, IEEE 802.3, popular numită **Ethernet™**, este o reţea cu difuzare bazată pe magistrală cu control descentralizat, lucrând la viteze între 10 Mbps şi 10 Gbps. Calculatoarele dintr-un Ethernet pot transmite oricând doresc; dacă două sau mai multe pachete se ciocnesc, fiecare calculator aşteaptă o perioadă de timp aleatorie şi apoi încearcă din nou.

Un al doilea tip de reţea cu difuzare este reţeaua în inel. Într-un inel fiecare bit se propagă independent de ceilalţi, fără să aştepte restul pachetului căruia îi aparţine. În mod tipic, fiecare bit navighează pe circumferinţa întregului inel într-un interval de timp în care se transmit doar câţiva biţi, de multe ori înainte chiar ca întregul pachet să fi fost transmis. Ca în orice alt sistem cu difuzare, este nevoie de o regulă pentru a arbitra accesele simultane la inel. Pentru aceasta se utilizează diferite metode, care vor fi discutate în carte mai târziu. IEEE 802.5 (inelul cu jeton de la IBM) este un LAN popular de tip inel, care operează la 4 şi la 16 Mbps. Un alt exemplu de reţea de tip inel este **FDDI** (**Fiber Distributed Data Interface**, rom: Interfaţă de date distribuite pe fibră optică).

Reţelele cu difuzare pot fi în continuare împărţite în statice şi dinamice, în funcţie de modul de alocare al canalului. O metodă tipică de alocare statică ar fi să divizăm timpul în intervale discrete şi să rulăm un algoritm round-robin, lăsând fiecare maşină să emită numai atunci când îi vine rândul. Alocarea statică iroseşte capacitatea canalului atunci când o maşină nu are nimic de transmis în cuanta de timp care i-a fost alocată, astfel că majoritatea sistemelor încearcă să aloce canalul dinamic (la cerere).

Metodele de alocare dinamică pentru un canal comun sunt fie centralizate, fie descentralizate. În cazul metodei centralizate de alocare a canalului există o singură entitate, de pildă o unitate de arbitrare a magistralei, care determină cine urmează la rând. Poate face acest lucru acceptând cereri şi luând o decizie conform unui algoritm intern. În cazul metodei descentralizate de alocare a canalului nu există o entitate centrală; fiecare maşină trebuie să hotărască pentru ea însăşi dacă să transmită sau nu. S-ar putea crede că în acest fel se ajunge totdeauna la haos,

dar lucrurile nu stau aşa. Vom studia mai târziu numeroşi algoritmi proiectaţi să refacă ordinea dintr-un potenţial haos.

1.2.2 Reţele metropolitane

O **reţea metropolitană** (**Metropolitan Area Network**), sau **MAN** (plural: MAN-uri) deserveşte un oraş. Cel mai bun exemplu de MAN este reţeaua de televiziune prin cablu disponibilă în cele mai multe oraşe. Acest sistem s-a dezvoltat de la primele antene colective folosite în zone în care semnalul recepţionat prin aer era foarte slab. În aceste sisteme timpurii, o antenă foarte mare era amplasată pe vârful celui mai apropiat deal şi semnalul captat era retransmis către casele abonaţilor.

La început, acestea erau sisteme proiectate local, ad-hoc. Apoi companiile au început să se implice în această afacere, obţinând contracte de la municipalităţile oraşelor pentru a cabla chiar şi întreg oraşul. Următorul pas a fost programarea televiziunii şi chiar canale de televiziune produse numai pentru furnizarea prin cablu. De cele mai multe ori aceste canale sunt foarte specializate, pe domenii precum ştirile, sporturile, gastronomia, grădinăritul, şi altele. Dar încă de la începuturi şi până în ultima perioadă a anilor 1990, aceste reţele erau exclusiv dedicate recepţiei de televiziune.

Din momentul în care Internet-ul a început să atragă audienţa de masă, operatorii de reţele de cablu TV au realizat că, dacă vor face anumite schimbări în sistem, ar putea să ofere servicii bidirecţionale în Internet în părţile nefolosite ale spectrului. La acel moment, sistemul de cablu TV a început să se transforme dintr-o soluţie de a distribui semnalul TV în oraş într-o reţea metropolitană. La o primă aproximare, o MAN poate să arate oarecum similar cu sistemul prezentat în fig. l-8.

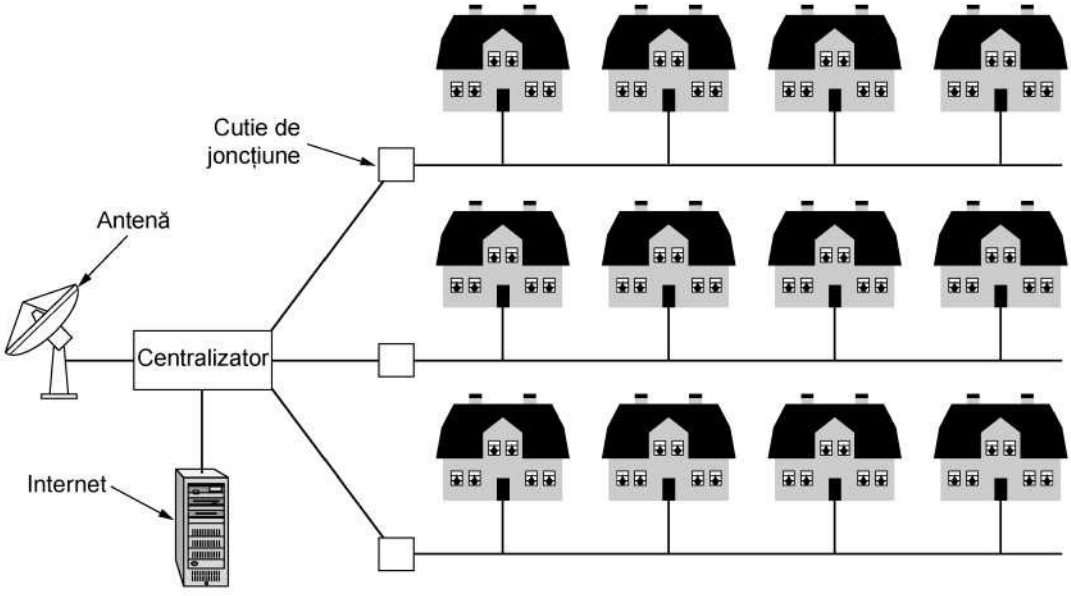


Fig. l-8. O reţea metropolitană care se bazează pe cablu TV.

În această figură se văd atât semnalele de televiziune cât şi Internet-ul trimise într-un centralizator (head end) pentru a fi apoi redistribuite în casele oamenilor. Vom reveni la acest subiect în detaliu în cap. 2.

Televiziunea prin cablu nu este singurul MAN. Ultimele dezvoltări în domeniul accesului la Internet fără fir, a dus la dezvoltarea unei noi reţele metropolitane care a fost standardizată cu numele de IEEE 802.16. Vom studia acest domeniu în cap. 2.

1.2.3 Reţele larg răspândite geografic

O **reţea larg răspândită geografic** (**Wide Area Network**), sau WAN, acoperă o arie geografică întinsă – deseori o ţară sau un continent întreg. Reţeaua conţine o colecţie de maşini utilizate pentru a executa programele utilizatorilor (adică aplicaţii). În concordanţă cu termenul uzual, vom numi aceste maşini **gazde**. Gazdele sunt conectate printr-o **subreţea de comunicaţie** sau, pe scurt, **subre- ţea**. Gazdele aparţin clienţilor (de exemplu calculatoarele personale ale oamenilor), deşi subreţeaua de comunicaţie aparţine şi este exploatată, de cele mai multe ori, de o companie de telefonie sau de un furnizor de servicii Internet (ISP). Sarcina subreţelei este să transporte mesajele de la gazdă la gazdă, exact aşa cum sistemul telefonic transmite cuvintele de la vorbitor la ascultător. Prin separarea aspectelor de pură comunicaţie ale reţelei (subreţelei) de aspectele referitoare la aplicaţii (gazde), proiectarea întregii reţele se simplifică mult.

În majoritatea reţelelor larg răspândite geografic, subreţeaua este formată din două componente distincte: liniile de transmisie şi elementele de comutare. **Liniile de transmisie** transportă biţii între maşini. Ele pot fi alcătuite din fire de cupru, fibră optică sau chiar legături radio. **Elementele de comutare** sunt calculatoare specializate, folosite pentru a conecta două sau mai multe linii de transmisie. Când sosesc date pe o anumită linie, elementul de comutare trebuie să aleagă o nouă linie pentru a retransmite datele mai departe. Din păcate, nu există nici o terminologie standard pentru denumirea acestor calculatoare. Aceste elemente de comutare au primit diverse nume în trecut; numele de **ruter** (**router**‡‡‡) este acum cel mai folosit.

În acest model, prezentat în fig. l-9, fiecare gazdă este de cele mai multe ori conectată la un LAN în care există un ruter, deşi în anumite cazuri o gazdă poate fi legată direct cu un ruter. Colecţia de linii de comunicaţie şi de rutere (dar nu şi gazdele) formează subreţeaua.

Subreţea Ruter

‡‡‡ Din păcate, unii îl pronunţă ca englezescul „rooter” şi alţii preferă să îl asocieze ca pronunţie cu „doubter”. Determinarea pronunţiei corecte în limba engleză va fi lăsată ca exerciţiu cititorului. (răspunsul pe care îl veţi afla poate depinde de zona în care întrebaţi).

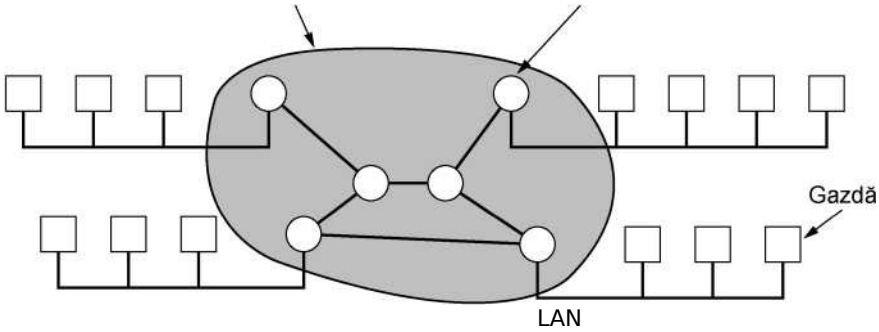


Fig. l-9. Relaţia dintre gazde şi subreţea.

Merită să facem un scurt comentariu în jurul termenului de „subreţea”. Iniţial, singura sa accepţiune se referea la colecţia ruterelor şi liniilor de comunicaţie care mutau pachetele de la gazda sursă la gazda destinaţie. Totuşi, câţiva ani mai târziu, cuvântul a mai căpătat un al doilea înţeles, în conjuncţie cu adresarea reţelelor (pe care o vom discuta în Cap. 5). Din nefericire, nu există o alternativă larg acceptată pentru înţelesul său iniţial, drept care noi vom folosi acest termen, cu unele rezerve, în ambele sensuri. Din context, va fi totdeauna clar care din ele este subînţeles.

În cazul celor mai multe WAN-uri, reţeaua conţine numeroase linii de transmisie, fiecare din ele legând o pereche de rutere. Dacă două rutere nu împart un acelaşi cablu, dar doresc să comunice, atunci ele trebuie să facă acest lucru indirect, prin intermediul altor rutere. Când un pachet este transmis de la un ruter la altul prin intermediul unuia sau mai multor rutere, pachetul este primit în întregime de fiecare ruter intermediar, este reţinut acolo până când linia de ieşire cerută devine liberă şi apoi este retransmis. O subreţea care funcţionează pe acest principiu se numeşte subreţea **me- morează-şi-retransmite** sau subreţea **cu comutare de pachete**. Aproape toate reţelele larg răspândite geografic (excepţie făcând cele care utilizează sateliţi) au subreţele memorează-şi-retransmite. Când pachetele sunt mici şi au aceeaşi mărime, ele sunt adesea numite **celule**.

Principiul de funcţionare a unui WAN cu comutare de pachete este atât de important încât merită să mai adăugăm câteva cuvinte despre el. În general, atunci când un proces al unei gazde are un mesaj de transmis către un proces de pe o altă gazdă, gazda care transmite va sparge mesajul în pachete, fiecare dintre ele reţinându-şi numărul de ordine din secvenţă. Aceste pachete sunt apoi transmise în reţea unul câte unul într-o succesiune rapidă. Pachetele sunt transportate individual prin reţea şi depozitate la gazda receptoare, unde sunt reasamblate în mesajul iniţial şi furnizate pro-

cesului receptor. Un flux de pachete rezultat din descompunerea unui mesaj iniţial oarecare este prezentat în fig. l-10.

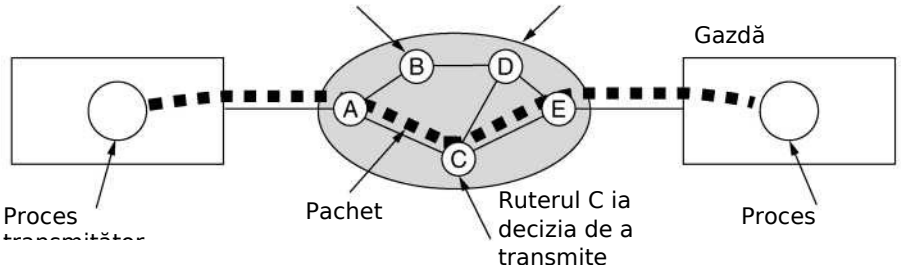


Fig. l-10. Un flux de pachete de la transmiţător la receptor.

În această figură, toate pachetele parcurg ruta A-C-E, în loc de A-B-D-E sau A-C-D-E. În unele reţele, toate pachetele aparţinând unui mesaj dat trebuie să urmeze aceeaşi rută; în altele, fiecare pachet este dirijat separat. Desigur, dacă A-C-E este cea mai bună rută, toate pachetele pot fi transmise pe acolo, chiar dacă fiecare dintre ele este dirijat individual.

Deciziile de dirijare se iau la nivelul local al ruterului. Când un pachet ajunge la ruterul A, este de datoria lui A să decidă dacă acest pachet trebuie trimis pe linia către B sau pe linia către C. Modul în care ruterul A ia această decizie este denumit **algoritm de rutare**. Există mulţi astfel de algoritmi. Pe unii dintre ei îi vom studia în detaliu în cap. 5.

Nu toate WAN-urile sunt cu comutare de pachete. O a doua posibilitate pentru un WAN este un sistem de sateliţi. Fiecare ruter are o antenă prin care poate trimite şi poate primi. Toate ruterele pot asculta ieşirea *de la* satelit, iar în anumite cazuri pot să asculte chiar şi transmisia celorlalte rutere *către* satelit. Uneori, ruterele sunt conectate la o reţea punct-la-punct şi numai unele dintre ele pot avea antene de satelit. Reţelele satelit sunt în mod implicit reţele cu difuzare şi sunt foarte utile când proprietatea de difuzare este importantă.

1.2.4 Reţele fără fir

Comunicaţiile digitale fără fir nu reprezintă o idee nouă. Încă din 1901, fizicianul italian Guglielmo Marconi a realizat legătura între un vapor şi un punct de pe coastă folosind telegraful fără fir şi codul Morse (punctele şi liniile sunt, în definitiv, binare). Sistemele radio moderne au performanţe mai bune, dar ideea fundamentală a rămas aceeaşi.

La o primă aproximare, reţelele fără fir pot fi împărţite în 3 mari categorii:

1. Interconectarea componentelor unui sistem
2. LAN-uri fără fir
3. WAN-uri fără fir

Interconectarea componentelor se referă numai la interconectarea componentelor unui calculator folosind unde radio cu rază mică de acţiune.

Aproape orice calculator are un monitor, o tastatură, un mouse şi o imprimantă legate la unitatea central ă prin cabluri. Mulţi dintre noii utilizatori au probleme cu conectarea tuturor cablurilor exact în mufele micuţe în care trebuie (chiar dacă acestea

sunt de cele mai multe ori codificate pe culori), aşa că producătorii de calculatoare oferă opţiunea de a trimite un tehnician pentru instalare. În consecinţă, câteva companii s-au adunat pentru a proiecta o reţea fără fir cu rază mică de acţiune denumită Bluetooth pentru a conecta toate aceste componente fără cabluri. De asemenea, Bluetooth permite camerelor digitale, căştilor, scanerelor şi altor dispozitive să se conecteze la calculator prin simpla poziţionare în zona acoperită de reţea. Fără cabluri, fără instalarea de drivere, doar poziţionare, pornire şi ... merge. Pentru mulţi oameni această uşurinţă în utilizare este un mare avantaj.

În cea mai simplă formă, reţelele de interconectare în sistem folosesc paradigma stăpân-sclav (master-slave) din fig. l-11(a). Unitatea centrală a sistemului este în mod normal stăpânul, care discută cu perifericele ca sclavi. Stăpânul le comunică sclavilor ce adrese să folosească, când pot să difuzeze mesaje, cât timp pot să transmită, ce frecvenţe pot să folosească, şi aşa mai departe. Vom discuta despre Bluetooth în detaliu în cap. 4.

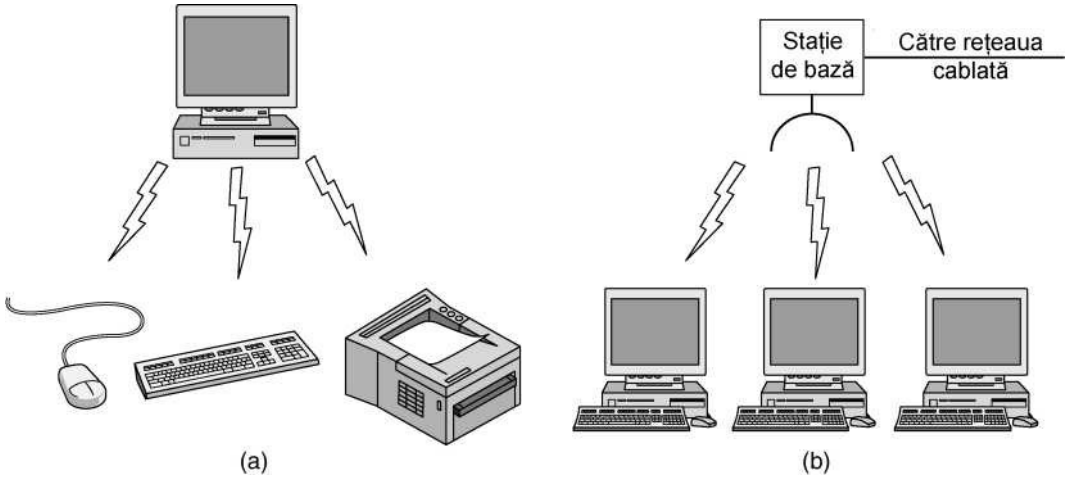


Fig. l-11. (a) Configuraţie Bluetooth. (b) Reţea locală

Următoarea treaptă în reţelele fără fir o reprezintă reţelele locale fără fir. Acestea sunt sisteme în care fiecare calculator are un modem radio şi o antenă cu care poate comunica cu alte calculatoare. De multe ori există o antenă în tavan cu care maşinile vorbesc, aşa cum se poate vedea în fig. 111 (b). Oricum, dacă sistemele sunt destul de apropiate, ele pot comunica direct unul cu altul într-o configuraţie punct-la-punct. Reţelele locale fără fir devin din ce în ce mai utilizate în birouri mai mici şi acasă, unde instalarea unei reţele Ethernet este considerată prea complicată, precum şi în clădiri de birouri mai vechi, în cantinele companiilor, în camerele de conferinţe, şi în alte asemenea locuri. Există un standard pentru reţele locale fără fir, numit **IEEE 802.11,** pe care îl implementează majoritatea sistemelor şi care devine din ce în ce mai răspândit. Îl vom discuta în cap. 4.

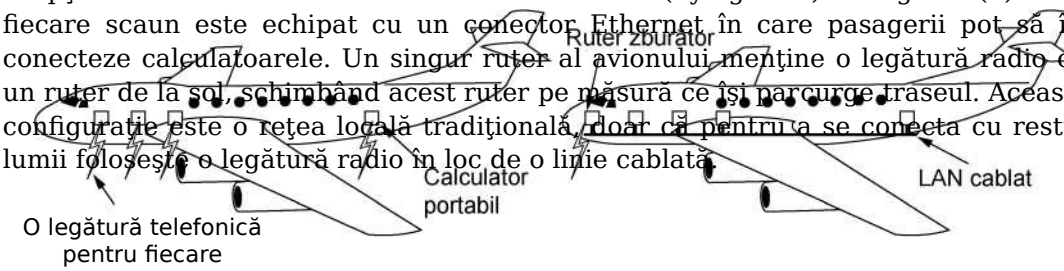
Cea de-a treia categorie de reţele fără fir este folosită în sistemele răspândite pe

arii geografice largi (Wide Area Networks). Reţeaua radio utilizată de telefonia mobilă este un exemplu de sistem fără fir cu lărgime de bandă redusă. Acest sistem este deja la generaţia a treia. Prima generaţie era analogică şi numai pentru voce. A doua generaţie era digitală, dar numai pentru voce. Cea de-a treia generaţie este digitală şi este utilizată atât pentru voce cât şi pentru date. Într-un anume sens, reţelele celulare fără fir sunt foarte asemănătoare cu reţelele locale fără fir, cu excepţia faptului că distan

ţele implicate sunt mult mai mari, iar ratele de transfer sunt mult mai mici. Reţelele locale fără fir pot opera la rate de până la 50 Mbps pe distanţe de zeci de metri. Sistemele celulare pot opera sub 1 Mbps, dar distanţele dintre staţia de bază şi calculator sau telefon este măsurată mai degrabă în kilometri decât în metri. Vom avea multe de spus despre aceste reţele în cap. 2.

În plus faţă de aceste reţele de viteză redusă, sunt dezvoltate şi WAN-uri cu lărgime de bandă mare. Important este în primul rând accesul la Internet de acasă sau din cadrul companiei prin conexiune rapidă fără fir, eliminând necesitatea folosirii sistemului de telefonie. Acest serviciu este de multe ori denumit serviciu local de distribuire multipunct. Îl vom studia mai târziu în carte. A fost dezvoltat şi un standard al său, numit IEEE 802.16. Îl vom examina în cap. 4.

Aproape toate reţelele ajung mai devreme sau mai târziu să fie parte dintr-o reţea cablată pentru a oferi acces la fişiere, baze de date sau Internet. Sunt multe variante prin care aceste conexiuni pot fi realizate, în funcţie de circumstanţe. De exemplu, în fig. l-12(a) este prezentat un avion în care un număr de persoane folosesc modemuri şi telefoane încorporate în spătarul scaunului (eng.: seatback telephone) pentru a suna la birou. Fiecare apel este independent de toate celelalte. O opţiune mult mai eficientă este LAN-ul zburător (flying LAN) din fig. l-12(b). Aici, fiecare scaun este echipat cu un conector Ethernet în care pasagerii pot să îşi conecteze calculatoarele. Un singur ruter al avionului menţine o legătură radio cu un ruter de la sol, schimbând acest ruter pe măsură ce îşi parcurge traseul. Această configuraţie este o reţea locală tradiţională, doar că pentru a se conecta cu restul lumii foloseşte o legătură radio în loc de o linie cablată.



(a) (b)

Fig. l-12. (a) Calculatoare mobile individuale. (b) Un LAN zburător.

Multă lume crede că tehnologiile fără fir reprezintă valul viitorului (de ex. Bi et al., 2001; Leeper, 2001; Varshez şi Vetter, 2000), dar exisită cel puţin o părere contrară cunoscută. Bob Metcalfe, inventatorul Ethernet-ului, a scris următoarele: ,,Calculatoarele mobile fără fir sunt ca băile mobile fără ţevi – ni şte oliţe de noapte portabile. Ele vor fi ceva comun în vehicule, pe şantiere şi la concerte rock. Sfatul meu este să vă racordaţi cabluri în casă şi să rămâneţi acolo” (Metcalfe, 1995). Istoria ar putea să reţină această afirmaţie în aceeaşi categorie cu a lui T.J. Watson, preşedintele IBM, care explica în 1945 de ce IBM nu se intră în afacerea calculatoarelor: „Patru sau cinci calculatoare ar trebui să fie suficiente pentru întreaga lume până în anul 2000”.

1.2.5 Reţelele casnice (Home networks)

Reţelele în mediul casnic sunt la orizont. Ideea fundamentală este că în viitor, cele mai multe locuinţe vor fi pregătite pentru instalarea de reţele. Fiecare dispozitiv din casă va fi capabil să comunice cu orice alt dispozitiv şi toate vor fi accesibile prin Internet. Acesta este unul dintre acele concepte

revoluţionare pe care nu l-a cerut nimeni (cum sunt telecomenzile TV sau telefoanele mobile), dar de îndată ce au fost implementate nimeni nu şi-a mai putut închipui cum au trăit fără ele.

Multe dispozitive sunt capabile să fie legate în reţea. Unele dintre categoriile cele mai simple, însoţite de exemple sunt cele care urmează:

1. Calculatoarele (PC-uri staţionare, PC-uri portabile, PDA-uri, periferice partajate)
2. Dispozitivele de divertisment (TV, DVD, VCR, camera video, combina muzicală)
3. Dispozitive pentru telecomunicaţii (telefonul, telefonul mobil, fax-ul, sistemul de comunicare interioară)
4. Aparatura casnică (cuptorul cu microunde, frigiderul, ceasul, cuptorul, aparatul de aer condiţionat, luminile)
5. Contoarele şi alarmele (contoarele pentru utilităţi, alarmele de fum sau hoţi, termosta- tele, sistemele de supraveghere a copilului)

Reţelele casnice sunt deja implementate într-o oarecare măsură. Multe case au deja un dispozitiv pentru conectarea mai multor calculatoare la Internet printr-o conexiune rapidă. Divertismentul prin reţea nu este chiar la îndemână, dar pentru că din ce în ce mai multă muzică şi mai multe filme sunt disponibile pentru descărcare din Internet, va exista o cerere de conectare a combinelor muzicale şi a televizoarelor în reţea. De asemenea, oamenii vor dori să împartă propriile clipuri video cu prietenii şi familia, astfel că această conexiune va trebui să fie bidirecţională. Angrenajul telecomunicaţiilor este deja conectat la lumea exterioară, dar în curând aceste vor fi digitale şi transmise prin Internet. În medie, o casă are cam o duzină de ceasuri (de exemplu, cele de la aparatele electrocas- nice), care toate trebuie potrivite cel puţin de doua ori pe an, când se trece la ora de vară şi apoi la ora de iarnă. Dacă toate aceste ceasuri ar fi conectate la Internet, această potrivire s-ar face automat. În fine, monitorizarea de la distanţă a casei şi a interiorului său este un posibil domeniu de succes. Probabil că mulţi dintre părinţi ar fi gata să cheltuiască nişte bani pentru a-şi supraveghea copiii adormiţi, prin intermediul PDA-urilor, în timp ce iau masa în oraş, chiar şi dacă au angajat un adolescent pentru a avea grijă de ei. În timp ce unii îşi pot imagina o reţea separată pentru fiecare zonă de aplicaţii, integrarea tuturor într-o singură reţea mai mare este probabil o idee mult mai bună.

Reţelele casnice au câteva proprietăţi fundamental diferite de alte tipuri de reţele. Mai întâi, atât reţeaua cât şi dispozitivele trebuie să fie uşor de instalat. Autorul a instalat multe componente hardware şi software pe diverse calculatoare de-a lungul anilor, cu diverse rezultate. O serie de telefoane la biroul de suport tehnic al producătorului au rezultat în răspunsuri de tipul (1) Citiţi manualul, (2) Reporniţi calculatorul, (3) Scoateţi toate componentele hardware şi software cu excepţia celor furnizate de noi şi încercaţi din nou, (4) Descărcaţi cea mai nouă versiune a programului de configurare de pe situl nostru Web şi dacă toate acestea eşuează, (5) Reformataţi discul şi apoi reinstalaţi Windows de pe CD-ROM. A spune unui cumpărător de frigider care poate fi conectat la Internet să descarce şi să

instaleze o nouă versiune a sistemului de operare pentru frigiderul său nu este de natură să facă prea mulţi clienţi fericiţi. Utilizatorii de calculatoare sunt obişnuiţi cu instalarea de produse care nu merg din prima; cumpărătorii de maşini, televizoare sau frigidere sunt mai puţin toleranţi. Ei se aşteaptă ca produsele să răspundă corect la 100% din comenzi.

În al doilea rând, reţelele şi dispozitivele trebuie să fie protejate împotriva utilizării neglijente. Primele aparate de aer condiţionat aveau un buton cu patru poziţii: OPRIT, SCĂZUT, MEDIU, RAPID. Acum au manuale de 30 de pagini. De îndată ce vor fi conectate în reţea, aşteptaţi-vă ca numai capitolul de securizare să aibă 30 de pagini. Ceea ce va depăşi capacitatea de înţelegere a majorităţii utilizatorilor.

În al treilea rând, preţul scăzut este esenţial pentru succes. Cumpărătorii nu vor plăti 50 de dolari în plus pentru un termostat numai pentru că unii oameni consideră important să-şi supravegheze de la birou temperatura din casă. Pentru numai 5 dolari în plus, s-ar putea să se vândă.

În al patrulea rând, programul principal este foarte probabil să implice facilităţi multimedia, aşa că reţeaua are nevoie de capacitate suficientă. Nu există piaţă pentru televizoare conectate la Internet care să prezinte filme de groază în rezoluţie de 320 x 240 pixeli şi la 10 cadre/s. Ethernet-ul rapid (fast Ethernet), mediul de lucru în majoritatea birourilor, nu este destul de bun pentru facilităţile multimedia. În consecinţă, reţelele casnice vor avea nevoie de performanţe mai bune decât cele are reţelelor care există acum în companii şi de preţuri mai mici pentru a deveni articole care se vând în masă.

În cel de-al cincilea rând, trebuie să fie posibil să se pornească cu unul sau două dispozitive şi extinderea să se poată face gradat. Aceasta înseamnă fără schimbări revoluţionare. A spune consumatorilor să îşi cumpere periferice cu interfeţe IEEE 1394 (FireWire) şi apoi, după câţiva ani, să retractezi spunând că USB 2.0 este interfaţa lunii va face consumatorii să devină capricioşi. Interfaţa de reţea va trebui să rămână stabilă pentru mulţi ani; cablajul (dacă există) va trebui să rămână acelaşi pentru decade întregi.

În cel de-al şaselea rând, securitatea şi siguranţa vor fi foarte importante. Pierderea câtorva fişiere datorită unui virus de poştă electronică e una, dar dacă un hoţ îţi dezarmează sistemul de securitate al locuinţei de la PDA-ul său şi apoi intră în casă este cu totul altă situaţie.

O întrebare interesantă este dacă reţelele casnice trebuie să fie cablate sau fără fir. Majoritatea locuinţelor au deja şase reţele instalate: electrică, telefonică, televiziune prin cablu, apă, gaz şi canalizare. Adăugarea unei a şaptea reţele în timpul construcţiei nu este dificilă, dar reamenajarea caselor deja construite este costisitoare. Costul este un motiv de a alege reţelele fără fir, dar securitatea este un motiv pentru cele cablate. Problema cu reţelele fără fir este aceea că undele radio pe care le folosesc trec foarte uşor prin garduri. Nimeni nu este foarte bucuros dacă vecinii îi pot intercepta conexiunea la Internet şi îi pot citi mesajele de poştă electronică în timp ce acestea sunt trimise la imprimantă. În cap. 8 vom vedea cum se poate folosi criptarea pentru a oferi securitate, dar în contextul unei reţele casnice, securitatea trebuie să fie şi ea protejată împotriva utilizării neglijente,

chiar şi în cazul utilizatorilor fără experienţă. Aceasta este mai uşor de spus decât de făcut, chiar şi pentru utilizatori foarte pricepuţi. Pe scurt, reţelele casnice oferă multe facilităţi şi provocări. Multe dintre ele sunt legate de necesitatea de a fi uşor de administrat, sigure şi securizate, mai ales în mâinile utilizatorilor care nu sunt implicaţi în domeniul tehnic, concomitent cu necesitatea de a obţine performanţe ridicate la preţuri scăzute.

1.2.6 Inter-reţelele

În lume există multe reţele, cu echipamente şi programe diverse. Persoanele conectate la o anumită reţea doresc adesea să comunice cu persoane racordate la alta. Această cerinţă impune conectarea unor reţele diferite, de multe ori incompatibile, ceea ce uneori se realizează utilizând maşini numite **porţi** (**gateways**). Acestea realizează conectarea şi asigură conversiile necesare, atât în termeni de hardware cât şi de software. O colecţie de reţele interconectate este numită **inter-reţea** sau **internet**. Aceşti termeni vor fi folosiţi în sens generic, spre deosebire de Internet-ul mondial (care este un internet special), al cărui nume va fi scris mereu cu majusculă.

O formă comună de inter-reţea este o colecţie de LAN-uri conectate printr-un WAN. De fapt, dacă am înlocui eticheta „subreţea” din fig. l-9 prin ,,WAN”, în figură nu ar mai trebui schimbat nimic altceva. În acest caz, singura diferenţă tehnică reală între o subreţea şi un WAN se referă la prezenţa gazdelor. Dacă sistemul din interiorul zonei gri conţine numai rutere, atunci este o subre- ţea. Dacă el conţine atât rutere, cât şi gazde cu utilizatori proprii, atunci este un WAN. Diferenţele reale sunt legate de proprietate şi utilizare.

Deseori se produc confuzii între subreţele, reţele şi inter-reţele. Termenul de subreţea este mai potrivit în contextul unei reţele larg răspândite geografic, unde se referă la colecţia de rutere şi linii de comunicaţie aflate în proprietatea operatorului de reţea. Ca o analogie, sistemul telefonic constă din centrale telefonice de comutare, care sunt conectate între ele prin linii de mare viteză şi sunt legate la locuinţe şi birouri prin linii de viteză scăzută. Aceste linii şi echipamente, deţinute şi întreţinute de către compania telefonică, formează subreţeaua sistemului telefonic. Telefoanele propriu- zise (care corespund în această analogie gazdelor) nu sunt o parte a subreţelei. Combinaţia dintre o subreţea şi gazdele sale formează o reţea. În cazul unui LAN, reţeaua este formată din cablu şi gazde. Aici nu există cu adevărat o subreţea.

O inter-reţea se formează atunci când se leagă între ele reţele diferite. Din punctul nostru de vedere, legarea unui LAN şi a unui WAN sau legarea a două LAN-uri formează o inter-reţea, dar nu există un consens asupra terminologiei din acest domeniu. O regulă simplă este aceea că dacă diferite companii sunt plătite să construiască diverse părţi ale unei reţele şi fiecare trebuie să îşi întreţină propria parte, avem o inter-reţea mai degrabă decât o singură reţea. De asemenea, dacă tehnologiile diferă în diverse zone ale reţelei (de exemplu: difuzare şi punct-la-punct), probabil că discutăm nu despre una ci despre două reţele.

1.3 PROGRAMELE DE REŢEA

În proiectarea primelor reţele de calculatoare, s-a acordat atenţie în primul rând echipamentelor, iar programele au fost gândite ulterior. Această strategie nu mai este valabilă. Programele de reţea sunt acum foarte structurate. În secţiunile următoare vom examina unele detalii ale tehnicii de structurare a programelor. Metoda descrisă aici formează punctul de sprijin al întregii cărţi şi ea va apărea mai departe în repetate rânduri.

1.3.1 Ierarhiile de protocoale

Pentru a reduce din complexitatea proiectării, majoritatea reţelelor sunt organizate sub forma unei serii de **straturi** sau **niveluri**, fiecare din ele construit peste cel de dedesubt. Numărul de niveluri, numele fiecărui nivel, conţinutul şi funcţia sa variază de la reţea la reţea. Oricum, în toate reţelele, scopul fiecărui nivel este să ofere anumite servicii nivelurilor superioare, protejându-le totodată de detaliile privitoare la implementarea efectivă a serviciilor oferite. Într-un anumit sens, fiecare nivel este un fel de maşină virtuală, oferind anumite servicii nivelului de deasupra lui.

Nivelul *n* de pe o maşină conversează cu nivelul *n* de pe altă maşină. Regulile şi convenţiile utilizate în conversaţie sunt cunoscute sub numele de **protocolul** nivelului *n.* În principal, un protocol reprezintă o înţelegere între părţile care comunică, asupra modului de realizare a comunicării. Ca o analogie, atunci când o femeie este prezentată unui bărbat, ea poate hotărî să-i întindă bărbatului mâna. La rândul său, bărbatul poate decide fie să-i strângă, fie să-i sărute mâna, decizie care depinde, să spunem, dacă femeia este o avocată americană care a venit la o întâlnire de afaceri sau este o
prinţesă europeană prezentă la un bal. Încălcarea protocolului va face comunicarea mai dificilă, dacă nu chiar imposibilă.

În fig. l-13 este ilustrată o reţea cu cinci niveluri. Entităţile din niveluri corespondente de pe maşini diferite se numesc **egale**. Entităţile egale pot fi procese, dispozitive hardware, sau chiar fiinţe umane. Cu alte cuvinte, entităţile egale sunt cele care comunică folosind protocolul.

Gazda 1 **Gazda 2**

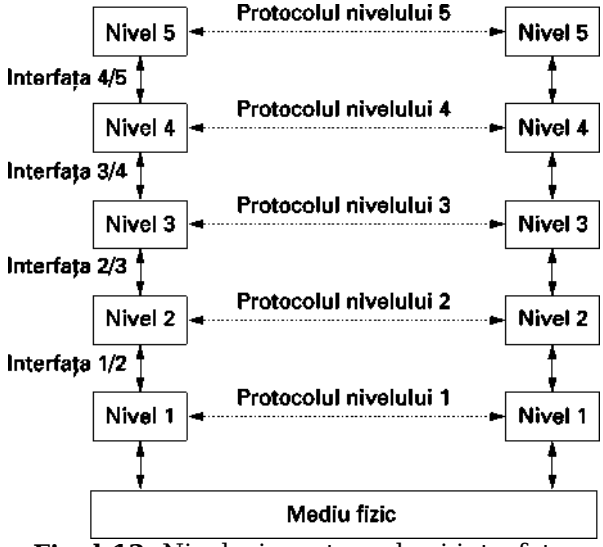


Fig. l-13. Niveluri, protocoale şi interfeţe.

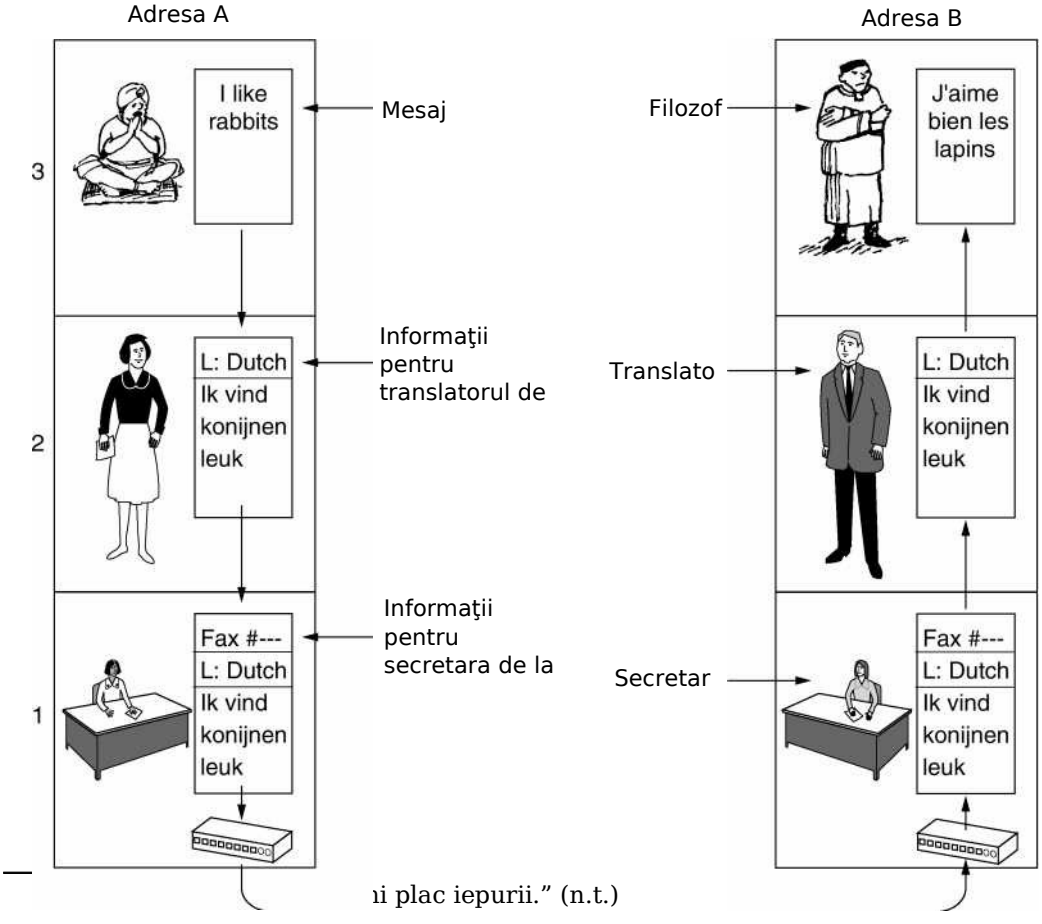
În realitate, nici un fel de date nu sunt transferate direct de pe nivelul *n* al unei maşini pe nivelul *n* al altei maşini. Fiecare nivel transferă datele şi informaţiile de control nivelului imediat inferior, până când se ajunge la nivelul cel mai de jos. Sub nivelul 1 se află **mediul fizic** prin care se produce comunicarea efectivă. În fig. l-13, comunicarea virtuală este reprezentată prin linii punctate, iar comunicarea fizică prin linii continue. Între două niveluri adiacente există o **interfaţă**. Interfaţa defineşte ce operaţii şi servicii primitive oferă nivelul de jos către nivelul de sus. Când proiectanţii de reţea decid câte niveluri să includă într-o reţea şi ce are de făcut fiecare din ele, unul din considerentele cele mai importante se referă la definirea de interfeţe clare între niveluri.

Aceasta presupune ca, la rândul său, fiecare nivel să execute o colecţie specifică de funcţii clar definite. Pe lângă minimizarea volumului de informaţii care trebuie transferate între niveluri, interfeţele clare permit totodată o mai simplă înlocuire a implementării unui nivel cu o implementare complet diferită (de exemplu, toate liniile telefonice se înlocuiesc prin canale de satelit). Aşa ceva este posibil, pentru că tot ceea ce i se cere noii implementări este să furnizeze nivelului superior exact setul de servicii pe care îl oferea vechea implementare. De altfel, este un fapt obişnuit ca doua gazde să folosească implementări diferite.

O mulţime de niveluri şi protocoale este numită **arhitectură de reţea**. Specificaţia unei arhitecturi trebuie să conţină destule informaţii pentru a permite unui proiectant să scrie programele sau să construiască echipamentele necesare fiecărui nivel, astfel încât nivelurile să îndeplinească corect protocoalele corespunzătoare. Nici detaliile de implementare şi nici specificaţiile interfeţelor nu fac parte din arhitectură, deoarece acestea sunt ascunse în interiorul maşinilor şi nu sunt vizibile din afară. Nu este necesar nici măcar ca interfeţele de pe maşinile dintr-o reţea să fie aceleaşi – cu condi

ţia, însă, ca fiecare maşină să poată utiliza corect toate protocoalele. O listă de protocoale utilizate de către un anumit sistem, câte un protocol pentru fiecare nivel, se numeşte **stivă de protocoale.** Arhitecturile de reţea, stivele de protocoale şi protocoalele propriu-zise constituie principalele subiecte ale acestei cărţi.

O analogie poate ajuta la explicarea ideii de comunicare multinivel. Imaginaţi-vă doi filosofi (procesele egale de la nivelul 3), unul din ei vorbind limbile urdu şi engleză, iar celălalt vorbind chineza şi franceza. Deoarece filosofii nu cunosc o limbă comună, fiecare din ei angajează câte un translator (procesele egale de la nivelul 2), iar fiecare translator contactează la rândul său o secretară (procesele egale de la nivelul 1). Filosoful 1 doreşte să comunice partenerului afecţiunea sa pentru *oryctolagus cuniculus.* Pentru aceasta, el trimite un mesaj (în engleză) prin interfaţa 2/3 către translatorul său, căruia îi spune următoarele cuvinte: „I like rabbits”[§§§] (ceea ce este ilustrat în fig. l-14). Translatorii s-au înţeles asupra unei limbi neutre, olandeza, aşa că mesajul este convertit în „Ik vind konijnen leuk.” Alegerea limbii reprezintă protocolul nivelului 2 şi este la latitudinea proceselor pereche de pe acest nivel.



§§§ ıi plac iepurii.” (n.t.)

Fig. l-14. Arhitectura filosof-translator-secretară.

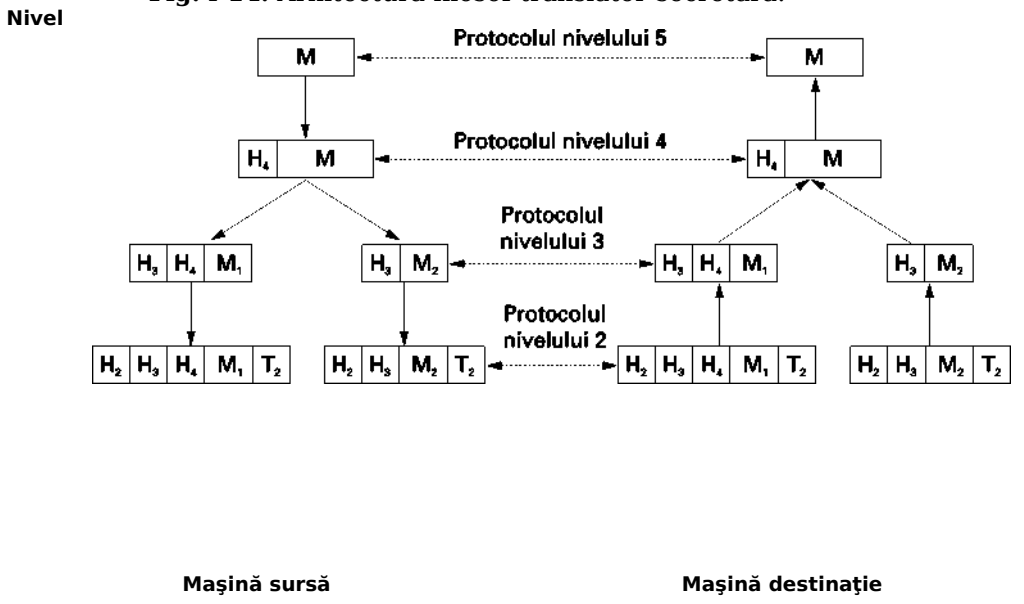


Fig. l-15. Exemplu de flux de informaţii pentru suportul comunicării virtuale la nivelul 5.

În continuare, translatorul înmânează mesajul secretarei, care îl trimite, de exemplu, prin fax (protocolul nivelului 1). Când mesajul este primit, el este tradus în franceză şi trimis prin interfaţa 2/3 către filosoful 2. Observaţi că, atâta timp cât interfeţele nu se modifică, fiecare protocol este complet independent de celelalte. Dacă doresc, translatorii pot schimba olandeza cu altă limbă, să spunem finlandeza, cu condiţia ca amândoi să se înţeleagă asupra acestui lucru şi ca nici unul din ei să nu îşi modifice interfaţa cu nivelul 1 sau cu nivelul 3. În mod similar, secretarele pot înlocui faxul cu poşta electronică sau cu telefonul fără a deranja (sau măcar a informa) celelalte niveluri. Fiecare proces poate adăuga anumite informaţii suplimentare destinate numai procesului său pereche. Aceste informaţii nu sunt transmise în sus, către nivelul superior.

Să considerăm acum un exemplu mai tehnic: cum se realizează comunicarea la ultimul nivel din reţeaua cu cinci niveluri din fig. l-15. O aplicaţie care se execută în nivelul 5 produce un mesaj M şi îl furnizează nivelului 4 pentru a-l transmite. Nivelul 4 inserează un **antet** în faţa mesajului, pentru a identifica respectivul mesaj şi pasează rezultatul nivelului 3. Antetul include informaţii de control, de exemplu numere de ordine care ajută nivelul 4 de pe maşina de destinaţie să livreze mesajele în ordinea corectă în cazul în care nivelurile inferioare nu păstrează această ordine. Pe unele niveluri, antetele conţin de asemenea câmpuri de control pentru mărime, timp şi alte informaţii.

În numeroase reţele nu există nici o limită cu privire la mărimea mesajelor transmise în protocolul nivelului 4, dar există aproape întotdeauna o limită impusă de protocolul nivelului 3. În consecinţă, nivelul 3 trebuie să spargă mesajele primite în unităţi mai mici, pachete, ataşând fiecărui pachet un antet specific nivelului 3. În

acest exemplu, M este descompus în două părţi, M_1 şi M_2.

Nivelul 3 decide ce linie de transmisie să utilizeze şi trimite pachetele nivelului 2. Nivelul 2 adaugă nu numai câte un antet pentru fiecare bucată, ci şi o încheiere, după care furnizează unitatea rezultantă nivelului 1 pentru a o transmite fizic. În maşina receptoare mesajul este trimis în sus, din nivel în nivel, pe parcurs fiind eliminate succesiv toate antetele. Nici un antet corespunzător nivelurilor de sub *n* nu este transmis în sus nivelului *n.*

Ceea ce este important de înţeles în fig. l-15 este relaţia dintre comunicaţia virtuală şi cea efectivă şi diferenţa între protocoale şi interfeţe. De exemplu, procesele egale de la nivelul 4 îşi imaginează conceptual comunicarea ca realizându-se pe „orizontală", utilizând protocolul nivelului 4. Deşi fiecare din ele are, probabil, o procedură de genul *TrimiteÎnCealaltăParte* şi o alta *PrimeşteDinCea- laltăParte,* aceste proceduri nu comunică de fapt cu cealaltă parte, ci cu nivelurile inferioare prin interfaţa 3/4.

Abstractizarea proceselor pereche este crucială pentru proiectarea întregii reţele. Cu ajutorul ei, această sarcină practic imposibilă poate fi descompusă în probleme de proiectare mai mici, rezolvabile, şi anume proiectarea nivelurilor individuale.

Deşi Secţiunea l-3 este intitulată „Programele de reţea", merită să subliniem că nivelurile inferioare dintr-o ierarhie de protocoale sunt implementate frecvent în hardware sau în firmware. Nu e mai puţin adevărat că aici intervin algoritmi complecşi, chiar dacă ei sunt înglobaţi (parţial sau în totalitate) în hardware.

1.3.2 Probleme de proiectare a nivelurilor

O parte din problemele cheie care apar la proiectarea reţelelor de calculatoare sunt prezente în mai multe niveluri. Vom menţiona pe scurt unele probleme mai importante.

Fiecare nivel are nevoie de un mecanism pentru a identifica emiţătorii şi receptorii. Dat fiind că o reţea cuprinde în mod normal numeroase calculatoare, iar o parte dintre acestea deţin mai multe procese, este necesară o modalitate prin care un proces de pe o anumită maşină să specifice cu cine doreşte să comunice. Ca o consecinţă a destinaţiilor multiple, pentru a specifica una dintre ele, este necesară o formă de adresare.

Un alt set de decizii de proiectare se referă la regulile pentru transferul de date. În unele sisteme datele circulă într-un singur sens; în altele datele pot circula în ambele sensuri. Protocolul trebuie, de asemenea, să determine câtor canale logice le corespunde conexiunea şi care sunt priorităţile acestora. Multe reţele dispun de cel puţin două canale logice pe conexiune, unul pentru date normale şi unul pentru date urgente.

Controlul erorilor este o problemă importantă deoarece circuitele fizice de comunicaţii nu sunt perfecte. Se cunosc multe coduri detectoare şi corectoare de erori, dar ambele capete ale conexiunii trebuie să se înţeleagă asupra codului utilizat. În plus, receptorul trebuie să aibă cum să-i spună emiţătorului care mesaje au fost primite corect şi care nu.

Nu toate canalele de comunicaţii păstrează ordinea mesajelor trimise. Pentru a putea trata o eventuală pierdere a secvenţialităţii, protocolul trebuie să furnizeze explicit receptorului informaţia necesară pentru a putea reconstitui mesajul. O soluţie evidentă este numerotarea fragmentelor, dar această soluţie încă nu rezolvă problema fragmentelor care sosesc la receptor aparent fără legătură cu restul mesajului.

O problemă ce intervine la fiecare nivel se referă la evitarea situaţiei în care un emiţător rapid trimite unui receptor lent date la viteză prea mare. Au fost propuse diverse rezolvări şi ele vor fi discutate mai târziu. Unele dintre acestea presupun o anumită reacţie, directă sau indirectă, prin care receptorul îl informează pe emiţător despre starea sa curentă. Altele limitează viteza de transmisie a emiţătorului la o valoare stabilită de comun acord cu receptorul. Acest subiect se numeşte **controlul fluxului**.

O altă problemă care apare la câteva niveluri priveşte incapacitatea tuturor proceselor de a accepta mesaje de lungime arbitrară. Acest fapt conduce la mecanisme pentru a dezasambla, a transmite şi apoi a reasambla mesajele. O problemă asemănătoare apare atunci când procesele insistă să transmită datele în unităţi atât de mici, încât transmiterea lor separată este ineficientă. În această situaţie, soluţia este să se asambleze împreună mai multe mesaje mici destinate aceluiaşi receptor şi să se dezasambleze la destinaţie mesajul mare obţinut astfel.

Atunci când este neconvenabil sau prea costisitor să se aloce conexiuni separate pentru fiecare pereche de procese comunicante, nivelul implicat în comunicare poate hotărî să utilizeze aceeaşi conexiune pentru mai multe conversaţii independente. Atâta timp cât această **mutiplexare** şi **demul- tiplexare** se realizează transparent, ea poate fi utilizată de către orice nivel. Multiplexarea este necesară, de exemplu, în nivelul fizic, unde traficul pentru toate conexiunile trebuie să fie transmis prin cel mult câteva circuite fizice.

Atunci când există mai multe căi între sursă şi destinaţie, trebuie ales un anumit drum. Uneori această decizie trebuie împărţită pe două sau mai multe niveluri. De exemplu, este posibil ca trimiterea unor date de la Londra la Roma să necesite atât o decizie la nivel înalt pentru alegerea ca ţară de tranzit a Franţei sau a Germaniei – în funcţie de legile lor de protejare a secretului datelor – cât şi o decizie de nivel scăzut pentru alegerea unuia din multele trasee posibile, pe baza traficului curent. Acest subiect poartă numele de **dirijare** sau **rutare** (**routing**).

1.3.3 Servicii orientate pe conexiuni şi servicii fără conexiuni

Nivelurile pot oferi nivelurilor de deasupra lor două tipuri de servicii: orientate pe conexiuni şi fără conexiuni. În această secţiune vom arunca o privire asupra acestor două tipuri şi vom examina diferenţele între ele.

Serviciul orientat pe conexiuni este modelat pe baza sistemului telefonic. Când vrei să vorbeşti cu cineva, mai întâi ridici receptorul, apoi formezi numărul, vorbeşti şi închizi. Similar, pentru a utiliza un serviciu orientat pe conexiuni, beneficiarul trebuie mai întâi să stabilească o conexiune, să folosească această conexiune şi apoi să o elibereze. În esenţă conexiunea funcţionează ca o ţeavă:

emiţătorul introduce obiectele (biţii) la un capăt, iar receptorul le scoate afară, în aceeaşi ordine, la celălalt capăt. În majoritatea cazurilor ordinea este menţinută, astfel încât biţii să ajungă în aceeaşi ordine în care au fost trimişi.

În anumite cazuri când se stabileşte o conexiune, transmiţătorul, receptorul şi subreţeaua negociază parametrii care vor fi folosiţi, cum sunt dimensiunea maximă a mesajului, calitatea impusă a serviciilor, şi alte probleme de acest tip. De obicei, una dintre părţi face o propunere şi cealaltă parte poate să o accepte, să o rejecteze sau să facă o contrapropunere.

Serviciul fără conexiuni este modelat pe baza sistemului poştal. Toate mesajele (scrisorile) conţin adresele complete de destinaţie şi fiecare mesaj circulă în sistem independent de celelalte. În mod normal, atunci când două mesaje sunt trimise la aceeaşi destinaţie, primul expediat este primul care ajunge. Totuşi, este posibil ca cel care a fost expediat primul să întârzie şi să ajungă mai repede al doilea. În cazul unui serviciu orientat pe conexiuni, aşa ceva este imposibil.

Fiecare serviciu poate fi caracterizat printr-o **calitate a serviciului**. Unele servicii sunt sigure în sensul că nu pierd date niciodată. De obicei, un serviciu sigur se implementează obligând receptorul să confirme primirea fiecărui mesaj, astfel încât expeditorul să fie sigur că mesajul a ajuns la destinaţie. Procesul de confirmare introduce un timp suplimentar şi întârzieri. Aceste dezavantaje sunt adesea acceptate, însă uneori ele trebuie evitate.

Transferul de fişiere este una din situaţiile tipice în care este adecvat un serviciu sigur orientat pe conexiuni. Proprietarul fişierului doreşte să fie sigur că toţi biţii ajung corect şi în aceeaşi ordine în care au fost trimişi. Foarte puţini utilizatori ai transferului de fişiere ar prefera un serviciu care uneori amestecă sau pierde câţiva biţi, chiar dacă acest serviciu ar fi mult mai rapid.

Serviciul sigur orientat pe conexiuni admite două variante: secvenţele de mesaje şi fluxurile de octeţi. Prima variantă menţine delimitarea între mesaje. Când sunt trimise două mesaje de 1024 de octeţi, ele vor sosi sub forma a două mesaje distincte de 1024 de octeţi, niciodată ca un singur mesaj de 2048 de octeţi. În a doua variantă, conexiunea este un simplu flux de octeţi şi nu există delimitări între mesaje. Când receptorul primeşte 2048 de octeţi, nu există nici o modalitate de a spune dacă ei au fost trimişi sub forma unui mesaj de 2048 octeţi, a două mesaje de 1024 de octeţi sau a 2048 mesaje de câte 1 octet. Dacă paginile unei cărţi sunt expediate unei maşini fotografice de tipărit printr-o reţea, sub formă de mesaje, atunci delimitarea mesajelor poate fi importantă. Pe de altă parte, în cazul unui utilizator care se conectează la un server aflat la distanţă, este nevoie numai de un flux de octeţi de la calculatorul utilizatorului la server. Delimitarea mesajelor nu mai este relevantă.

Aşa cum am menţionat mai sus, întârzierile introduse de confirmări sunt inacceptabile pentru unele aplicaţii. O astfel de aplicaţie se referă la traficul de voce digitizată. Pentru abonaţii telefonici este preferabil să existe puţin zgomot pe linie sau să audă ocazional câte un cuvânt distorsionat decât să se producă o întârziere din cauza aşteptării confirmării. Similar, atunci când se transmite o video- conferinţă, câţiva pixeli diferiţi nu reprezintă o problemă, în schimb

întreruperile pentru a corecta erorile ar fi extrem de supărătoare.

Nu orice aplicaţie necesită conexiuni. De exemplu, în măsura în care poşta electronică devine ceva tot mai uzual, se poate să nu apară foarte curând publicitatea prin poştă electronică? Expeditorul de publicitate prin poştă electronică probabil că nu vrea să se complice stabilind şi apoi eliberând o conexiune doar pentru un singur mesaj. Nici furnizarea la destinaţie cu o rată de corectitudine de 100% nu este esenţială, mai ales dacă lucrul acesta costă mai mult. Tot ceea ce se cere este un mijloc de a trimite un singur mesaj cu o probabilitate mare de a ajunge la destinaţie, dar fără o garanţie în acest sens. Serviciul nesigur (adică neconfirmat) fără conexiuni este deseori numit **serviciu datagramă**, prin analogie cu serviciul de telegrame – care, la rândul său, nu prevede trimiterea unei confirmări către expeditor.

În alte situaţii, avantajul de a nu fi necesară stabilirea unei conexiuni pentru a trimite un mesaj scurt este de dorit, dar siguranţa este de asemenea esenţială. Aceste aplicaţii pot utiliza **serviciul datagramă confirmat**. Este ca şi cum ai trimite o scrisoare recomandată şi ai solicita o confirmare de primire. În clipa în care soseşte confirmarea, expeditorul este absolut sigur că scrisoarea a fost livrată la destinaţia corectă şi nu a fost pierdută pe drum.

Mai există un serviciu, şi anume **serviciul cerere-răspuns**. În acest serviciu emiţătorul transmite o singură datagramă care conţine o cerere; replica primită de la receptor conţine răspunsul. În această categorie intră, de exemplu, un mesaj către biblioteca locală în care se întreabă unde este vorbită limba Uighur. Serviciul cerere-răspuns este utilizat în mod frecvent pentru a implementa comunicarea în modelul client-server: clientul lansează o cerere şi serverul răspunde la ea. În fig. l-16 sunt rezumate tipurile de servicii discutate mai sus.

Conceptul de a utiliza comunicaţii nesigure poate părea derutant la început. La urma urmei, de ce ar prefera cineva comunicaţiile nesigure în locul comunicaţiilor sigure? Mai întâi, comunicaţiile sigure (ceea ce înseamnă, pentru noi, confirmate) pot să nu fie disponibile. De exemplu, Ethernet-ul nu oferă comunicaţii sigure. Pachetele pot fi uneori alterate în timpul tranzitului. Urmează ca protocoalele nivelurilor superioare să se ocupe de această problemă.

Serviciu	Exemplu
Flux de mesaje sigur	Secvenţă de pagini
Flux de octeţi sigur	Conectare la distanţă
Conexiune nesigură	Voce digitizată
Datagramă nesigură	Publicitate prin e-mail
Datagramă confirmată	Scrisori cu confirmare
Cerere-răspuns	Interogări baze de date

În al doilea rând, întârzierile inerente în cazul în care se oferă servicii sigure ar putea fi inacceptabile, mai ales în cazul aplicaţiilor de timp real cum sunt aplicaţiile multimedia. Pentru aceste motive, comunicaţiile sigure cât şi cele nesigure coexistă.

Orientate pe conexiuni

Fără conexiuni

Fig. l-16. Şase tipuri diferite de servicii.

1.3.4 Primitive de serviciu

Un serviciu este specificat formal printr-un set de **primitive** (operaţii) puse la dispoziţia utilizatorului care foloseşte serviciul. Aceste primitive comandă serviciului să execute anumite acţiuni sau să raporteze despre acţiunile executate de o entitate pereche. Dacă stiva de protocoale este localizată în sistemul de operare, aşa cum se întâmplă de cele mai multe ori, primitivele sunt în mod normal apeluri sistem. Aceste apeluri cauzează o trecere a sistemului de operare în modul nucleu (kernel), care preia controlul maşinii pentru a trimite pachetele necesare.

Setul de primitive disponibile depinde de natura serviciului oferit. Primitivele serviciilor orientate pe conexiuni sunt diferite de cele ale serviciilor fără conexiuni. Ca un exemplu minimal de primitive de serviciu care pot fi oferite pentru a implementa un flux de octeţi într-un mediu client-server, putem considera primitivele listate în fig. l-17.

Primitiva	Semnificaţia
LISTEN (Ascultă)	Blocare în aşteptarea unei conexiuni
CONNECT (Conectează)	Stabilirea unei conexiuni cu o entitate pereche aflată în aşteptare
RECEIVE (Primeşte)	Blocare în aşteptarea unui mesaj

SEND (Trimite)	Trimite un mesaj entităţii pereche
DISCONNECT (Deconectează)	Termină o conexiune

Fig. l-17. Cinci primitive de serviciu pentru implementarea unui serviciu simplu orientat pe conexiune.

Aceste primitive pot fi folosite în următorul mod: mai întâi serverul execută LISTEN pentru a indica faptul că este pregătit să accepte conexiuni. Un mod obişnuit de a implementa LISTEN este a

face un apel de sistem blocant. După execuţia primitivei, procesul server este blocat până la apariţia unei cereri de conectare.

Apoi procesul client execută CONNECT pentru a stabili o conexiune cu serverul. Apelul CONNECT trebuie să specifice cu cine se doreşte conectarea, aşa că ar putea avea un parametru prin care se transmite adresa serverului. De cele mai multe ori, sistemul de operare va trimite un prim pachet entităţii pereche cerându-i să se conecteze, după cum este arătat de (1) în fig. l-18. Procesul client este suspendat până când apare un răspuns. Când pachetul ajunge la server, el este procesat de sistemul de operare al acestuia. Când sistemul de operare observă că pachetul cere o conexiune, verifică dacă există vreun ascultător. Dacă da, va face doua lucruri: va debloca ascultătorul şi va trimite înapoi o confirmare (2). Sosirea acestei confirmări eliberează apoi clientul. În acest moment, atât clientul cât şi serverul sunt în execuţie şi au stabilit o conexiune între ei. Este important de observat că secvenţa de confirmare (2) este generată de codul protocolului însuşi, nu ca răspuns al unei primitive de la nivelul utilizatorului. Dacă apare o cerere de conexiune şi nu există nici un ascultător, rezultatul este nedefinit. În anumite sisteme, pachetul poate fi păstrat un scurt timp într-o coadă, anticipând o eventuală comandă LISTEN.

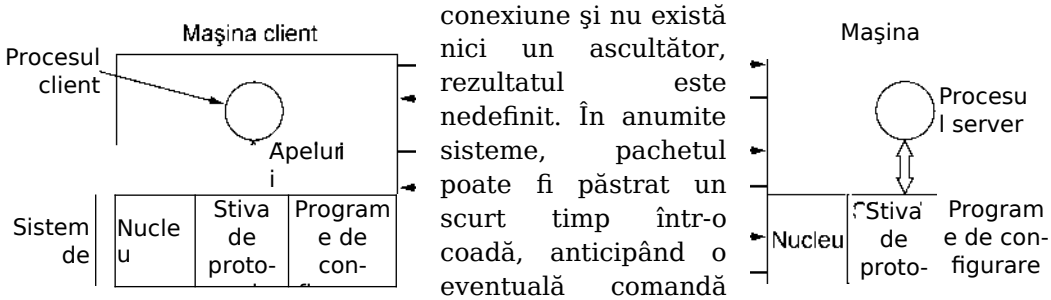


Fig. l-18. Pachetele trimise într-o simplă interacţiune client-server pe o reţea orientată pe conexiuni.

Analogia evidentă între acest protocol şi viaţa reală este cazul clientului care sună la directorul departamentului de service al unei companii. Directorul stă lângă telefon pentru a putea răspunde în cazul în care acesta sună. Clientul face un apel. Când directorul ridică receptorul, conexiunea este stabilită.

Pasul următor este ca serverul să execute RECEIVE pentru a se pregăti să accepte prima cerere. În mod normal serverul face această operaţie de îndată ce a fost eliberat din blocarea impusă de LISTEN, înainte să ajungă confirmarea înapoi la client. Apelul RECEIVE blochează serverul.

Apoi clientul execută SEND pentru a transmite cererea sa (3) urmat de execuţia unui RECEIVE pentru a obţine răspunsul.

Sosirea pachetului de cerere la maşina server deblochează procesul server astfel încât acesta să poată procesa cererea. După ce a terminat lucrul, foloseşte SEND pentru a răspunde clientului (4). Sosirea acestui pachet deblochează clientul care poate acum să analizeze răspunsul obţinut. Daca mai există cereri din partea clientului, acesta le poate face acum. Dacă a terminat, poate folosi DISCONNECT pentru a termina conexiunea. De obicei, apelul iniţial DISCONNECT este blocant, suspendând clientul şi trimiţând un pachet către server pentru a-i comunica faptul ca respectiva conexiune nu mai este necesară (5). Când serverul primeşte pachetul, el lansează un DISCONNECT propriu, confirmând cererea clientului şi eliberând conexiunea. Când pachetul serverului (6) ajunge înapoi la maşina clientului, procesul client este eliberat şi conexiunea este întreruptă. Foarte pe scurt, aşa funcţionează comunicaţiile orientate pe conexiuni.

Desigur, viaţa nu este simplă. Multe dintre lucruri pot să nu funcţioneze corect. Sincronizarea poate fi proastă (de exemplu, daca se încearcă un CONNECT înainte de LISTEN), pachetele se pot pierde şi multe altele. Vom studia toate acestea în detaliu ceva mai târziu, dar deocamdată fig. l-18 rezumă pe scurt modul în care ar putea să funcţioneze o comunicaţie client-server într-o reţea orientată pe conexiuni.

Ştiind că acele şase pachete sunt necesare pentru a realiza acest protocol, cititorul se poate întreba de ce nu se foloseşte un protocol fără conexiune în locul său. Răspunsul este că ar fi posibil într-o lume perfectă, şi atunci ar fi nevoie de numai două pachete: unul pentru cerere şi unul pentru răspuns. Oricum, în cazul real cu mesaje lungi în oricare dintre direcţii (de exemplu un fişier de 1 MB), cu erori de transmisie şi cu pachete pierdute, situaţia se modifică. Dacă răspunsul ar avea sute de pachete, dintre care unele s-ar putea pierde în timpul transmisiei, cum ar putea clientul să îşi dea seama că unele piese lipsesc? Cum ar putea şti clientul dacă ultimul pachet recepţionat este de fapt ultimul pachet trimis? S ă presupunem că de la client se face o cerere pentru un al doilea fişier. Cum ar putea clientul să diferenţieze pachetele din cel de-al doilea fişier de eventualele pachete pierdute din primul fişier? Pe scurt, în lumea reală, un simplu protocol cerere-răspuns implementat într-o reţea nesigură este de cele mai multe ori inadecvat. În cap. 3 vom studia în detaliu o largă varietate de protocoale, care pot rezolva aceste probleme şi altele similare. Pentru moment însă este de ajuns să spunem că a avea un flux de octeţi sigur şi ordonat între procese este de multe ori foarte convenabil.

1.3.5 Relaţia dintre servicii şi protocoale

Deşi sunt adesea confundate, serviciile şi protocoalele reprezintă concepte distincte. Diferenţa între ele este atât de importantă, încât o subliniem din nou în această secţiune. Un *serviciu* este un set de primitive (operaţii) pe care un nivel le furnizează nivelului de deasupra sa. Serviciul defineşte ce operaţii este pregătit nivelul să realizeze pentru utilizatorii săi, dar nu spune nimic despre cum sunt implementate aceste operaţii. Un serviciu este definit în contextul unei interfeţe între două niveluri, nivelul inferior fiind furnizorul serviciului şi nivelul superior

fiind utilizatorul serviciului.

Prin contrast, un *protocol* este un set de reguli care guvernează formatul şi semnificaţia cadrelor, pachetelor sau mesajelor schimbate între ele de entităţile pereche dintr-un nivel. Entităţile folosesc protocoale pentru a implementa definiţiile serviciului lor. Ele sunt libere să îşi schimbe protocoalele după cum doresc, cu condiţia să nu modifice serviciul pe care îl văd utilizatorii. În acest fel, serviciul şi protocolul sunt complet decuplate.

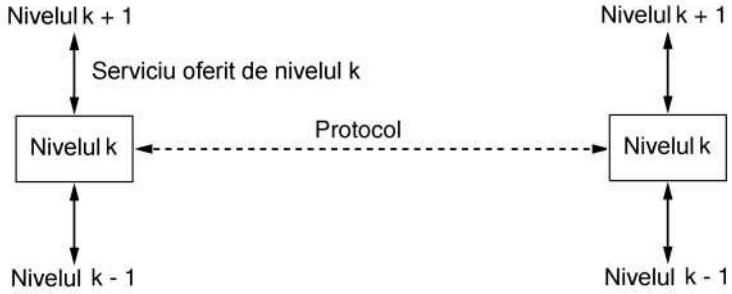


Fig. l-19. Relaţia dintre un server şi un protocol.

Cu alte cuvinte, serviciile sunt legate de interfeţele dintre niveluri, după cum este ilustrat şi în fig. l-19. Prin contrast, protocoalele sunt legate de pachetele trimise între entităţile pereche de pe diferite maşini. Este important să nu existe confuzii între cele două concepte.

Merită să facem o analogie cu limbajele de programare. Un serviciu este ca un tip de date abstracte sau ca un obiect într-un limbaj orientat pe obiecte. Acesta defineşte operaţiile care pot fi aplicate pe un obiect, dar nu specifică modul de implementare a operaţiilor. Un protocol se referă la *implementarea* serviciului şi nu este vizibil pentru utilizatorul serviciului.

Multe protocoale mai vechi nu făceau diferenţa între serviciu şi protocol. Ca urmare, un nivel tipic putea avea o primitivă de serviciu SEND PACKET în care utilizatorul furniza o referinţă către un pachet complet asamblat. Acest aranjament însemna că toate modificările protocolului erau imediat vizibile pentru utilizatori. Majoritatea proiectanţilor de reţele privesc acum un astfel de mecanism ca pe o eroare gravă.

1.4 MODELE DE REFERINŢĂ

Acum, după ce am discutat la modul abstract structura pe niveluri a reţelelor, a sosit timpul să studiem câteva exemple. În următoarele două secţiuni vom discuta două arhitecturi de reţea importante, modelul de referinţă OSI şi modelul de referinţă TCP/IP. Deşi *protocoalele* asociate cu modelul OSI nu sunt folosite aproape deloc, *modelul* în sine este destul de general şi încă valabil, iar caracteristicile puse în discuţie la fiecare nivel sunt în continuare foarte importante. Modelul TCP/IP are caracteristici opuse: modelul în sine nu este foarte util, dar

protocoalele sunt folosite pe scară largă. Din acest motiv, le vom studia pe fiecare în detaliu. În plus, uneori poţi învăţa mai multe din eşecuri decât din succese.

1.4.1 Modelul de referinţă OSI

Modelul OSI este prezentat în fig. l-16 (mai puţin mediul fizic). Acest model se bazează pe o propunere dezvoltată de către Organizaţia Internaţională de Standardizare (International Standards Organization – ISO) ca un prim pas către standardizarea internaţională a protocoalelor folosite pe diferite niveluri (Day şi Zimmermann, 1983). A fost revizuit în 1995 (Day, 1995). Modelul se numeşte **ISO OSI** (**Open Systems Interconnection**, rom: interconectarea sistemelor deschise), pentru că el se ocupă de conectarea sistemelor deschise – adică de sisteme deschise comunicării cu alte sisteme. În continuare vom folosi mai ales termenul prescurtat de model OSI.

Modelul OSI cuprinde şapte niveluri. Principiile aplicate pentru a se ajunge la cele şapte niveluri sunt următoarele:

1. Un nivel trebuie creat atunci când este nevoie de un nivel de abstractizare diferit.
2. Fiecare nivel trebuie să îndeplinească un rol bine definit.
3. Funcţia fiecărui nivel trebuie aleasă acordându-se atenţie definirii de protocoale standardizate pe plan internaţional.
4. Delimitarea nivelurilor trebuie făcută astfel încât să se minimizeze fluxul de informaţii prin interfeţe.

5. Numărul de niveluri trebuie să fie suficient de mare pentru a nu fi nevoie să se introducă în
acelaşi nivel funcţii diferite şi suficient de mic pentru ca arhitectura să rămână funcţională.

În continuare vom discuta fiecare nivel al modelului, începând cu nivelul cel mai de jos. Modelul OSI nu reprezintă în sine o arhitectură de reţea, pentru că nu specifică serviciile şi protocoalele utilizate la fiecare nivel. Modelul spune numai ceea ce ar trebui să facă fiecare nivel. ISO a produs de asemenea standarde pentru fiecare nivel, însă aceste standarde nu fac parte din modelul de referinţă propriu-zis. Fiecare din standardele respective a fost publicat ca un standard internaţional separat.

Nivelul fizic

Nivelul fizic se ocupă de transmiterea biţilor printr-un canal de comunicaţie. Proiectarea trebuie să garanteze că atunci când unul din capete trimite un bit 1, acesta e receptat în cealaltă parte ca un bit 1, nu ca un bit 0. Problemele tipice se referă la câţi volţi trebuie utilizaţi pentru a reprezenta un 1 şi câţi pentru un 0, dacă transmisia poate avea loc simultan în ambele sensuri, cum este stabilită conexiunea iniţială şi cum este întreruptă când au terminat de comunicat ambele părţi, câţi pini are conectorul de reţea şi la ce foloseşte fiecare pin. Aceste aspecte de proiectare au o legătură strânsă cu interfeţele mecanice, electrice, funcţionale şi

procedurale, ca şi cu mediul de transmisie situat sub nivelul fizic.

Nivelul legătură de date

Sarcina principală a **nivelului legăturii de date** este de a transforma un mijloc oarecare de transmisie într-o linie care să fie disponibilă nivelului reţea fără erori de transmisie nedetectate. Nivelul legătură de date realizează această sarcină obligând emiţătorul să descompună datele de intrare în **cadre de date** (în mod tipic, câteva sute sau câteva mii de octeţi) şi să transmită cadrele secvenţial. Dacă serviciul este sigur, receptorul confirmă fiecare cadru trimiţând înapoi un cadru de confirmare pozitivă.

O altă problemă care apare la nivelul legătură de date (şi, de asemenea, la majoritatea nivelurilor superioare) este evitarea inundării unui receptor lent cu date provenite de la un emiţător rapid. În acest scop sunt necesare mecanisme de reglare a traficului care să permită emiţătorului să afle cât spaţiu tampon deţine receptorul la momentul curent. Controlul traficului şi tratarea erorilor sunt deseori integrate. Reţelele cu difuzare determină în nivelul legătură de date o problemă suplimentară: cum să fie controlat accesul la canalul partajat. De această problemă se ocupă un subnivel special al nivelului legătură de date şi anume subnivelul de control al accesului la mediu.

Nivelul reţea

Nivelul reţea se ocupă de controlul funcţionării subreţelei. O problemă cheie în proiectare este determinarea modului în care pachetele sunt dirijate de la sursă la destinaţie. Dirijarea se poate baza pe tabele statistice care sunt „cablate” intern în reţea şi care sunt schimbate rar. Traseele pot fi de asemenea stabilite la începutul fiecărei conversaţii, de exemplu la începutul unei sesiuni la terminal (de ex. o operaţie de login pe o maşină la distanţă). În sfârşit, dirijarea poate fi foarte dinamică, traseele determinându-se pentru fiecare pachet în concordanţă cu traficul curent din reţea.

Dacă în subreţea există prea multe pachete simultan, ele vor intra unul pe traseul celuilalt şi astfel se vor produce gâtuiri. Controlul unor astfel de congestii îi revine tot nivelului reţea. Mai general, calitatea serviciilor oferite (întârziere, timp de tranzitare, fluctuaţii, etc.) este tot o responsabilitate a nivelului reţea.

Multe probleme pot apărea când un pachet trebuie să călătorească dintr-o reţea în alta ca să ajungă la destinaţie. Modul de adresare folosit de a doua reţea poate să difere de cel pentru prima. A doua reţea poate chiar să nu accepte deloc pachetul pentru că este prea mare. De asemenea, protocoalele pot fi diferite şi aşa mai departe. Rezolvarea acestor probleme în vederea interconectării reţelelor eterogene este sarcina nivelului reţea. În reţelele cu difuzare, problema dirijării este simplă, astfel că nivelul reţea este deseori subţire sau chiar nu există deloc.

Nivelul transport

Rolul principal al nivelului transport este să accepte date de la nivelul sesiune, să le descompună, dacă este cazul, în unităţi mai mici, să transfere aceste unităţi nivelului reţea şi să se asigure că toate fragmentele sosesc corect la celălalt capăt. În plus, toate acestea trebuie făcute eficient şi într-un mod care izolează nivelurile de mai sus de inevitabilele modificări în tehnologia echipamentelor.

Nivelul transport determină, de asemenea, ce tip de serviciu să furnizeze nivelului sesiune şi, în final, utilizatorilor reţelei. Cel mai obişnuit tip de conexiune transport este un canal punct-la-punct fără erori care furnizează mesajele sau octeţii în ordinea în care au fost trimişi. Alte tipuri posibile de servicii de transport sunt transportul mesajelor individuale – fără nici o garanţie în privinţa ordinii de livrare – şi difuzarea mesajelor către destinaţii multiple. Tipul serviciului se determină când se stabileşte conexiunea. (Ca un comentariu secundar: este imposibil de obţinut un canal fără erori; ceea ce oamenii înţeleg prin această expresie este că rata erorilor este destul de mică pentru a fi ignorată în practică).

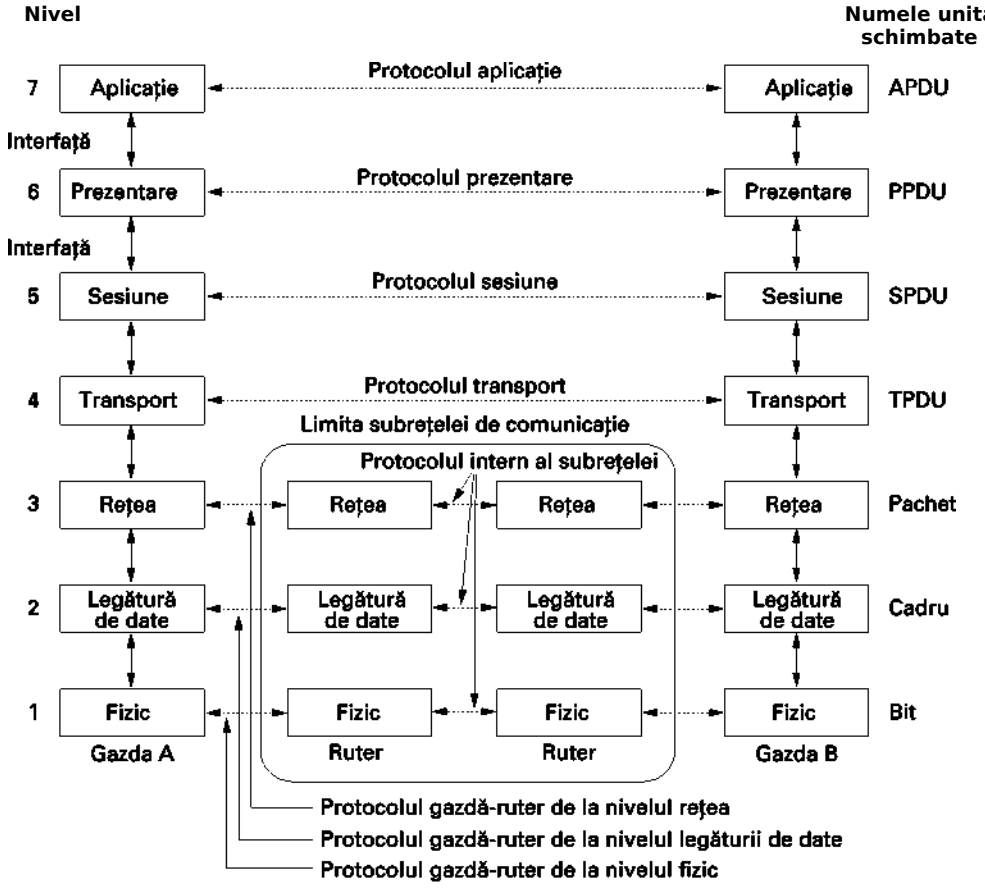


Fig. 1-20. Modelul de referinţă OSI.

Nivelul transport este un adevărat nivel capăt-la-capăt, de la sursă la destinaţie. Cu alte cuvinte, un program de pe maşina sursă poartă o conversaţie cu un program similar de pe maşina destinaţie, folosind în acest scop antetele mesajelor şi mesaje de control. În nivelurile inferioare protocoalele au loc între fiecare maşină şi vecinii săi imediaţi (niveluri înlănţuite), şi nu direct între maşinile sursă şi destinaţie (niveluri capăt-la-capăt), care pot fi separate de numeroase rutere. Diferenţa între nivelurile de la 1 până la 3, care sunt înlănţuite şi nivelurile de la 4 la 7, care sunt capăt-la-capăt, este ilustrată în fig. l-20.

Nivelul sesiune

Nivelul sesiune permite utilizatorilor de pe maşini diferite să stabilească între ei sesiuni. Sesiunile oferă diverse servicii, incluzând controlul dialogului (respectarea ordinii în raport cu dreptul de a transmite), **gestionarea jetonului** (prevenirea situaţiei în care două entităţi încearcă aceeaşi operaţie critică în acelaşi timp) şi **sincronizarea** (introducerea de puncte de control pe parcursul transmisiilor lungi, astfel încât, în cazul unui eşec, acestea să poată fi reluate de unde rămăseseră).

Nivelul prezentare

În particular, spre deosebire de nivelurile inferioare, care se ocupă numai de transferul biţilor dintr-un loc în altul, nivelul prezentare se ocupă de sintaxa şi semantica informaţiilor transmise. Pentru a face posibilă comunicarea între calculatoare cu reprezentări diferite ale datelor, structurile de date care se schimba între ele pot fi definite într-un mod abstract, alături de o codificare standardizată ce va fi utilizată „pe cablu”. Nivelul prezentare gestionează aceste structuri de date abstracte şi permite definirea şi comunicarea unor structuri de date de nivel mai înalt (de ex. înregistrări bancare).

Nivelul aplicaţie

Nivelul aplicaţie conţine o varietate de protocoale frecvent utilizate. Un exemplu de protocol utilizat pe scară largă este **HTTP** (**HyperText Transfer Protocol**, rom: protocol de transfer al hiper- textului), care sta la baza **WWW** (**World Wide Web**, rom: reţea de întindere planetară). Atunci când un program de navigare (browser) accesează o pagină Web, el trimite serverului numele paginii pe care o doreşte folosind HTTP. Serverul va trimite ca răspuns pagina. Alte protocoale de aplicaţie sunt folosite pentru transferul fişierelor, poştă electronica, ştiri în reţea.

1.4.2 Modelul de referinţă TCP/IP

Să ne îndreptăm acum atenţia de la modelul de referinţă OSI spre modelul de referinţă utilizat de strămoşul tuturor reţelelor de calculatoare, ARPANET-ul, şi de succesorul său, Internet-ul. Deşi vom prezenta mai târziu o scurtă istorie a ARPANET-ului, este util să menţionăm acum câteva aspecte esenţiale. ARPANET a fost o reţea de cercetare sponsorizată de către DoD (U.S. Department of Defense, rom: Departamentul de Apărare al Statelor Unite). În cele din urmă, reţeaua a ajuns să conecteze între ele, utilizând linii telefonice închiriate, sute de reţele universitare şi guvernamentale. Atunci când au fost adăugate, mai târziu, reţele prin satelit şi radio, interconectarea acestora cu protocoalele existente a pus diferite probleme. Era nevoie de o nouă arhitectură de referinţă. De aceea, posibilitatea de a interconecta fără probleme mai multe tipuri de reţele a reprezentat de la bun început un obiectiv de proiectare major. Această arhitectură a devenit cunoscută mai târziu sub denumirea de **modelul de referinţă TCP/IP**, dată după numele celor două protocoale fundamentale utilizate. Arhitectura respectivă a fost definită prima dată în (Cerf şi Kahn, 1974). O perspectivă ul

terioară este prezentată în (Leiner ş.a., 1985). Filosofia de proiectare din spatele modelului este discutată în (Clark, 1988).

Dată fiind îngrijorarea Departamentului de Apărare că o parte din preţioasele sale gazde, rutere şi porţi de interconectare ar putea fi distruse dintr-un moment în altul, un alt obiectiv major a fost ca reţeaua să poată supravieţui pierderii echipamentelor din subreţea fără a fi întrerupte conversaţiile existente. Cu alte cuvinte, DoD dorea ca, atâta timp cât funcţionau maşina sursă şi maşina destinaţie, conexiunile să rămână intacte, chiar dacă o parte din maşini sau din liniile de transmisie erau brusc scoase din funcţiune. Mai mult, era nevoie de o arhitectură flexibilă, deoarece se aveau în vedere aplicaţii cu cerinţe divergente, mergând de la transferul de fişiere până la transmiterea vorbirii în timp real.

Nivelul internet

Toate aceste cerinţe au condus la alegerea unei reţele cu comutare de pachete bazată pe un nivel inter-reţea fără conexiuni. Acest nivel, numit **nivelul internet**, este axul pe care se centrează întreaga arhitectură. Rolul său este de a permite gazdelor să emită pachete în orice reţea şi a face ca pachetele să circule independent până la destinaţie (fiind posibil ca aceasta să se găsească pe o altă reţea). Pachetele pot chiar să sosească într-o ordine diferită faţă de cea în care au fost trimise, caz în care – dacă se doreşte livrarea lor ordonată – rearanjarea cade în sarcina nivelurilor superioare. De observat că „internet” este folosit aici într-un sens generic, chiar dacă acest nivel este prezent şi în Internet.

Aici, analogia este cu sistemul de poştă (clasică). O persoană dintr-o anumită ţară poate depune într-o cutie poştală mai multe scrisori internaţionale şi, cu puţin noroc, majoritatea scrisorilor vor ajunge la adresa corectă din ţara de destinaţie. Probabil că scrisorile vor trece pe drum prin mai multe oficii de cartare, dar acest lucru se face transparent pentru utilizatori. Mai mult, faptul că fiecare ţară (adică fiecare reţea) are propriile timbre, propriile mărimi favorite de plicuri şi propriile reguli de livrare este ascuns beneficiarilor.

Nivelul internet defineşte oficial un format de pachet şi un protocol numit **IP (Internet Protocol**, rom: protocol Internet). Sarcina nivelului internet este să livreze pachete IP către destinaţie. Problemele majore se referă la dirijarea pachetelor şi evitarea congestiei. În consecinţă, este rezonabil să spunem că nivelul internet din TCP/IP funcţionează asemănător cu nivelul reţea din OSI. Fig. l-21 arată această corespondenţă.

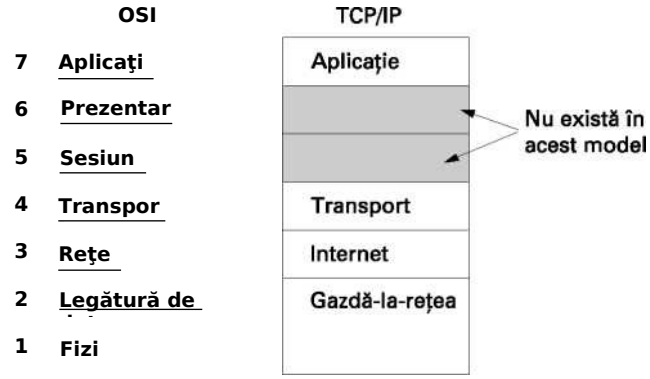


Fig. l-21. Modelul de referinţă TCP/IP.

Nivelul transport

Nivelul situat deasupra nivelului internet din modelul TCP/IP este frecvent numit **nivelul transport**. Acesta este proiectat astfel, încât să permită conversaţii între entităţile pereche din gazdele sursă şi, respectiv, destinaţie, la fel ca în nivelul transport OSI. În acest sens au fost definite două protocoale capăt-la-capăt. Primul din ele, **TCP** (**Transmission Control Protocol**, rom: protocolul de control al transmisiei), este un protocol sigur orientat pe conexiuni care permite ca un flux de octeţi trimişi de pe o maşină să ajungă fără erori pe orice altă maşină din inter-reţea. Acest protocol fragmentează fluxul de octeţi în mesaje discrete şi pasează fiecare mesaj nivelului internet. La destinaţie, procesul TCP receptor reasamblează mesajele primite într-un flux de ieşire. TCP tratează totodată controlul fluxului pentru a se asigura că un emiţător rapid nu inundă un receptor lent cu mai multe mesaje decât poate acesta să prelucreze.

Al doilea protocol din acest nivel, **UDP** (**User Datagram Protocol**, rom: protocolul datagramelor utilizator), este un protocol nesigur, fără conexiuni, destinat aplicaţiilor care doresc să utilizeze propria lor secvenţiere şi control al fluxului, şi nu pe cele asigurate de TCP. Protocolul UDP este de asemenea mult folosit pentru interogări rapide întrebare-răspuns, client-server şi pentru aplicaţii în care comunicarea promptă este mai importantă decât comunicarea cu acurateţe, aşa cum sunt aplicaţiile de transmisie a vorbirii şi a imaginilor video. Relaţia dintre IP, TCP şi UDP este prezentată în fig. l-22. De când a fost dezvoltat acest model, IP a fost implementat pe multe alte reţele.

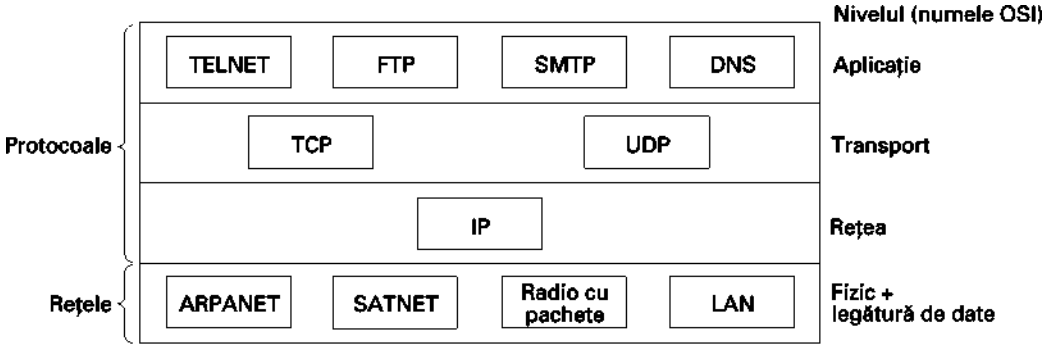


Fig. 1-22. Protocoale și rețele din modelul TCP/IP inițial.

Nivelul aplicaţie

Modelul TCP/IP nu conţine niveluri sesiune sau prezentare. Acestea nu au fost incluse pentru că nu s-a simţit nevoia lor. Experienţa modelului OSI a dovedit că această viziune a fost corectă: în majoritatea aplicaţiilor, nivelurile respective nu sunt de mare folos.

Deasupra nivelului transport se află **nivelul aplicaţie**. Acesta conţine toate protocoalele de nivel mai înalt. Aşa cum se vede din fig. l-22, primele protocoale de acest gen includeau terminalul virtual (TELNET), transferul de fişiere (FTP) şi poşta electronică (SMTP). Protocolul de terminal virtual permite unui utilizator de pe o maşină să se conecteze şi să lucreze pe o maşină aflată la distanţă. Protocolul de

transfer de fişiere pune la dispoziţie o modalitate de a muta eficient date de pe o maşină pe alta. Poşta electronică a fost la origine doar un tip de transfer de fişiere, dar ulterior a fost dezvoltat un protocol specializat (SMTP – Simple Mail Transfer Protocol, rom: Protocol simplu de transfer al poştei) pentru acest serviciu. Pe parcursul anilor, la aceste protocoale s-au adăugat multe altele, aşa cum sunt Serviciul Numelor de Domenii (Domain Name Service – DNS) pentru stabilirea corespondenţei dintre numele gazdelor şi adresele reţelelor, NNTP, protocolul utilizat pentru a transfera articole de ştiri USENET, H'l'l'P, folosit pentru aducerea paginilor de pe Web şi multe altele.

Nivelul gazdă-reţea

Sub nivelul internet se află necunoscutul. Modelul de referinţă TCP/IP nu spune mare lucru despre ce se întâmplă acolo, însă menţionează că gazda trebuie să se lege la reţea, pentru a putea trimite pachete IP, folosind un anumit protocol. Acest protocol nu este definit şi variază de la gazdă la gazdă şi de la reţea la reţea. Cărţile şi articolele despre TCP/IP rareori discută despre acest protocol.

1.4.3 O comparaţie între modelele de referinţă OSI şi TCP

Modelele de referinţă OSI şi TCP/IP au multe lucruri în comun. Amândouă se bazează pe conceptul unei stive de protocoale independente. De asemenea, funcţionalitatea nivelurilor este în linii mari similară. De exemplu, în ambele modele, nivelurile până la nivelul transport inclusiv sunt necesare pentru a pune la dispoziţia proceselor care doresc să comunice un serviciu de transport capăt-la-capăt independent de reţea. Nivelurile respective formează furnizorul de transport. Din nou, în ambele modele, nivelurile de deasupra transportului sunt beneficiari orientaţi pe aplicaţii ai serviciului de transport.

În pofida acestor similitudini fundamentale, între cele două modele există şi multe deosebiri. În această secţiune ne vom concentra asupra diferenţelor cheie dintre cele două modele de referinţă. Este important de subliniat că vom compara aici *modelele de referinţă,* nu *stivele de protocoale* corespunzătoare. Protocoalele propriu-zise vor fi discutate mai târziu. Pentru o întreagă carte consacrată comparaţiei şi diferenţelor dintre TCP/IP şi OSI, a se vedea (Piscitello şi Chapin, 1993).

Trei concepte sunt esenţiale pentru modelul OSI:

1. Servicii
2. Interfeţe
3. Protocoale

Probabil că cea mai mare contribuţie a modelului OSI este că a făcut explicită diferenţa între aceste trei concepte. Fiecare nivel realizează nişte servicii pentru nivelul situat deasupra sa. Definiţia *serviciului* spune ce face nivelul, nu cum îl folosesc entităţile de deasupra sa sau cum funcţionează nivelul. El defineşte semantica nivelului.

Interfaţa unui nivel spune proceselor aflate deasupra sa cum să facă accesul. Interfaţa precizează ce reprezintă parametrii şi ce rezultat se obţine. Nici interfaţa

nu spune nimic despre funcţionarea internă a nivelului.

În sfârşit, *protocoalele* pereche folosite într-un nivel reprezintă treaba personală a nivelului. Nivelul poate folosi orice protocol doreşte, cu condiţia ca acesta să funcţioneze (adică să îndeplinească serviciul oferit). Nivelul poate de asemenea să schimbe protocoalele după cum vrea, fără ca acest lucru să afecteze programele din nivelurile superioare.

Aceste idei se potrivesc foarte bine cu ideile moderne referitoare la programarea orientată pe obiect. Un obiect, ca şi un nivel, posedă un set de metode (operaţii) care pot fi invocate de către procese din afara obiectului. Semanticele acestor metode definesc mulţimea de servicii pe care le oferă obiectul. Parametrii şi rezultatele metodelor formează interfaţa obiectului. Codul intern al obiectului reprezintă protocolul său şi nu este vizibil şi nici important în afara obiectului.

Deşi lumea a încercat ulterior să îl readapteze pentru a fi mai asemănător modelului OSI, modelul TCP/IP nu a făcut iniţial o distincţie clară între serviciu, interfaţă şi protocol. De exemplu, singurele servicii veritabile oferite de nivelul internet sunt SEND IP PACKET şi RECEIVE IP PACKET.

În consecinţă, protocoalele din modelul OSI sunt mai bine ascunse decât în modelul TCP/IP şi pot fi înlocuite relativ uşor pe măsură ce se schimbă tehnologia. Capacitatea de a face asemenea modificări reprezintă unul din scopurile principale ale organizării protocoalelor pe niveluri în modelul OSI.

Modelul de referinţă OSI a fost conceput *înainte* să fie inventate protocoalele corespunzătoare. Ordinea respectivă semnifică faptul că modelul nu a fost orientat către un set specific de protocoale, fiind prin urmare destul de general. Reversul este că proiectanţii nu au avut multă experienţă în ceea ce priveşte acest subiect şi nu au avut o idee coerentă despre împărţirea funcţiilor pe niveluri.

De exemplu, nivelul legătură de date se ocupa iniţial numai cu reţelele punct-la-punct. Atunci când au apărut reţelele cu difuzare, a trebuit să fie introdus în model un subnivel nou. Când lumea a început să construiască reţele reale utilizând modelul OSI şi protocoalele existente, s-a descoperit că acestea nu se potriveau cu specificaţiile serviciului cerut (minunea minunilor), astfel că a trebuit introdusă în model convergenţa subnivelurilor, ca să existe un loc pentru a glosa pe marginea diferenţelor. În sfârşit, comitetul se aştepta iniţial ca fiecare ţară să aibă câte o reţea care să fie în custodia guvernului şi să folosească protocoalele OSI, aşa că nu s-a dat nici o atenţie interconectării. Pentru a nu mai lungi povestea, să spunem doar că lucrurile s-au petrecut altfel.

În ceea ce priveşte TCP/IP, lucrurile stau exact pe dos: mai întâi au apărut protocoalele, iar modelul a fost de fapt doar o descriere a protocoalelor existente. Cu protocoalele respective nu era nici o problemă: ele se potriveau perfect cu modelul. Singurul necaz era că *modelul* nu se potrivea cu nici o altă stivă de protocoale. Prin urmare, modelul nu a fost prea util pentru a descrie alte reţele non-TCP/IP.

Pentru a ne întoarce de la subiectele filosofice la subiecte mai specifice, o diferenţă evidentă între cele două modele se referă la numărul de niveluri: modelul OSI are şapte niveluri, iar TCP/IP are patru. Ambele modele au niveluri

(inter-)reţea, transport şi aplicaţie, dar restul nivelurilor sunt diferite.

O altă deosebire priveşte subiectul comunicaţiei fără conexiuni faţă de cel al comunicaţiei orientată pe conexiuni. Modelul OSI suportă ambele tipuri de comunicaţii la nivelul reţea, dar numai comunicaţii orientate pe conexiuni în nivelul transport, unde acest fapt are importanţă (pentru că serviciul de transport este vizibil utilizatorilor). Modelul TCP/IP are numai un mod (fără conexiuni) la nivelul reţea, dar suportă ambele moduri la nivelul transport, ceea ce lasă utilizatorilor posibilitatea alegerii. Această alegere este importantă în mod special pentru protocoale întrebare-răspuns simple.

1.4.4 O critică a modelului şi protocoalelor OSI

Nici modelul şi protocoalele OSI şi nici modelul şi protocoalele TCP/IP nu sunt perfecte. Asupra lor se pot formula, şi s-au formulat, câteva critici. În prezenta şi în următoarea secţiune vom vedea unele dintre aceste critici. Vom începe cu OSI, după care vom examina TCP/IP.

La momentul când a fost publicată a doua ediţie a acestei cărţi (1989), majoritatea experţilor în domeniu credeau că modelul şi protocoalele OSI se vor impune peste tot şi vor elimina orice concurent. Acest lucru nu s-a întâmplat. De ce? O privire spre lecţiile trecutului poate fi utilă. Aceste lecţii pot fi rezumate astfel:

1. Ratarea momentului.
2. Tehnologii proaste.
3. Implementări proaste.
4. Politici proaste.

Ratarea momentului

Să vedem mai întâi prima problemă: ratarea momentului. Momentul la care se stabileşte un standard este absolut critic pentru succesul acestuia. David Clark de la M.I.T. are o teorie asupra standardelor pe care o numeşte *Apocalipsa celor doi elefanţi* şi care este ilustrată în fig. l-23.

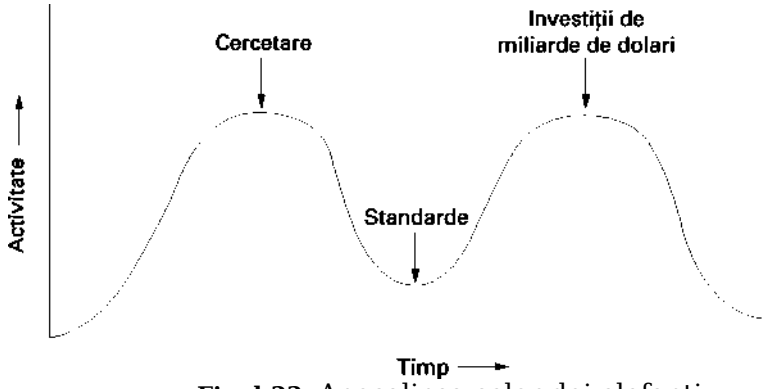


Fig. l-23. Apocalipsa celor doi elefanţi.

Această figură arată volumul de activitate desfăşurată în jurul unui subiect nou. Când subiectul este lansat, are loc o explozie a activităţii de cercetare sub formă de discuţii, articole şi întâlniri. După un timp, cercetarea se reduce foarte mult, subiectul este descoperit de companii şi piaţa cunoaşte un val de investiţii de miliarde de dolari.

Este esenţial ca standardele să fie definite în intervalul dintre cei doi „elefanţi". Dacă ele sunt definite prea devreme, înainte să se încheie cercetarea, atunci subiectul poate să nu fie încă destul de bine înţeles, ceea ce conduce la standarde proaste. Dacă ele sunt definite prea târziu, atunci probabil că atât de multe firme au făcut deja investiţii majore realizând lucrurile altfel, încât standardele sunt efectiv ignorate. Dacă intervalul dintre cei doi elefanţi este foarte scurt (pentru că toată lumea arde de nerăbdare să treacă la lucru), atunci cei care dezvoltă standardele pot fi prinşi la mijloc şi striviţi.

Acum se vede că protocoalele OSI standard au fost strivite. La momentul apariţiei lor, protocoalele concurente TCP/IP erau deja folosite pe scară largă în universităţi, în cercetare. Înainte să vină valul investiţiilor de miliarde de dolari, piaţa din domeniul academic era destul de dezvoltată pentru ca multe firme să înceapă, prudent, să ofere produse TCP/IP. Când a apărut OSI, firmele nu au mai vrut, decât forţate, să sprijine o a doua stivă de protocoale, şi, prin urmare, n-au apărut nici un fel de oferte iniţiale din partea lor. Fiecare firmă aştepta să înceapă celelalte firme, aşa că până la urmă n-a mai început nici o firmă şi fenomenul OSI nu s-a mai produs niciodată.

Tehnologii proaste

Al doilea motiv pentru care OSI n-a prins niciodată este că atât modelul cât şi protocoalele au defecte. Opţiunea pentru şapte niveluri a fost mai mult politică decât tehnică , şi două dintre niveluri (sesiune şi prezentare) sunt aproape goale, în timp ce alte două (legătura de date şi reţea) sunt prea aglomerate.

Modelul OSI, alături de protocoalele şi definiţiile de servicii asociate, este extraordinar de complex. Atunci când sunt puse unul peste altul, standardele tipărite au o grosime de câţiva zeci de centimetri. Standardele sunt, de asemenea, dificil de implementat şi ineficiente în funcţionare. În acest context îmi vine în minte o ghicitoare formulată de Paul Mockapetris şi citată în (Rose, 1993):

Î: Ce obţii când aplici un standard internaţional unui gangster?
R: O persoană care îţi face o ofertă pe care n-o poţi înţelege.

Pe lângă faptul că este incomprehensibil, o altă problemă cu OSI este că unele funcţii, cum sunt adresarea, controlul fluxului şi controlul erorilor apar repetat în fiecare nivel. Saltzer ş.a. (1994), de exemplu, au arătat că, pentru a fi eficient, controlul erorilor trebuie făcut la nivelul cel mai înalt şi că repetarea sa de atâtea ori în nivelurile de mai jos este adesea inutilă şi ineficientă.

Implementări proaste

Dată fiind enorma complexitate a modelului şi a protocoalelor, nu este de mirare în faptul că implementările iniţiale erau uriaşe, greoaie şi ineficiente. Oricine le încerca se simţea ca opărit. Nu a trecut mult şi lumea a asociat „OSI" cu „calitate slabă." Deşi odată cu trecerea timpului produsele au devenit mai bune, imaginea s-

a deteriorat.

Din contră, una din primele implementări de TCP/IP făcea parte din Berkeley UNIX şi era destul de bună (ca să nu mai spunem că era şi gratuită). Lumea a început să o folosească repede, ceea ce a determinat apariţia unei comunităţi largi de utilizatori, ceea ce a dus mai departe la îmbunătăţiri, iar aceasta a dus la o comunitate şi mai numeroasă. În acest caz spirala nu cobora, ci urca.

Politici proaste

Din cauza implementării iniţiale, multă lume, în special din mediul academic, a considerat TCP/IP ca o parte din Unix; iar în anii ‘80 Unix-ul era pentru oamenii din lumea academică cam la fel de popular ca paternitatea (numită apoi incorect maternitate) sau ca plăcinta cu mere.

OSI, pe de altă parte, a fost gândit ca o creaţie a ministerelor de telecomunicaţii europene, apoi a Comunităţii Europene şi, mai târziu, a guvernului Statelor Unite. Această viziune s-a dovedit adevărată numai în parte; dar chiar ideea în sine – un grup de birocraţi guvernamentali încercând să bage un standard inferior tehnic pe gâtul bieţilor cercetători şi programatori care stau în tranşee şi dezvoltă efectiv reţelele de calculatoare – nu a ajutat prea mult. Unii oameni au văzut această abordare în aceeaşi lumină în care a fost văzut IBM când a anunţat în anii ‘60 că PL/I era limbajul viitorului, sau DoD care a corectat IBM-ul anunţând că limbajul respectiv era de fapt Ada.

1.4.5 O critică a modelului de referinţă TCP/IP

Modelul şi protocoalele TCP/IP au şi ele problemele lor. Mai întâi, modelul nu face o distincţie clară între conceptele de serviciu, interfaţă şi protocol. O practică recomandabilă în ingineria programării este să se facă diferenţa între specificaţie şi implementare, ceea ce OSI face cu multă atenţie, pe când TCP/IP nu face. De aceea, modelul TCP/IP nu este un ghid prea bun de proiectare a reţelelor noi folosind tehnologii noi.

În al doilea rând, modelul TCP/IP nu este deloc general şi nu este aproape deloc potrivit pentru descrierea altor stive de protocoale în afara celei TCP/IP. De exemplu, descrierea Bluetooth folosind modelul TCP/IP ar fi aproape imposibilă.

În al treilea rând, nivelul gazdă-reţea nu este deloc un nivel – în sensul normal în care este folosit termenul în contextul protocoalelor organizate pe niveluri – ci este o interfaţă (între nivelurile reţea şi legătură de date). Distincţia între o interfaţă şi un nivel este crucială şi de aceea trebuie să i se acorde atenţia cuvenită.

În al patrulea rând, modelul TCP/IP nu distinge (şi nici măcar nu menţionează) nivelurile fizic şi legătură de date. Acestea sunt complet diferite. Nivelul fizic are de-a face cu caracteristicile transmisiei prin cablu de cupru, fibre optice sau radio. Rolul nivelului legătură de date este să delimiteze începutul şi sfârşitul cadrelor şi să le transporte dintr-o parte în alta cu gradul de siguranţă dorit. Un model corect ar trebui să includă ambele niveluri ca niveluri separate. Modelul TCP/IP nu face acest lucru.

În sfârşit, deşi protocoalele IP şi TCP au fost atent gândite şi bine implementate, multe din celelalte protocoale au fost construite ad-hoc, fiind în general opera

câtorva absolvenţi care tot „meştereau” la ele până oboseau. Implementările protocoalelor erau apoi distribuite gratuit; ca urmare, ele erau larg utilizate, fără să li se asigure suportul necesar, fiind de aceea greu de înlocuit. Unele protocoale au ajuns acum să fie mai mult o pacoste. Protocolul de terminal virtual, TELNET, de exemplu, a fost proiectat pentru un terminal teletype mecanic de zece caractere pe secundă. Cu toate acestea, 25 de ani mai târziu, protocolul este încă foarte utilizat.

Pentru a rezuma, în pofida acestor probleme, *modelul* OSI (mai puţin nivelurile sesiune şi prezentare) s-a dovedit a fi excepţional de util pentru a discuta reţelele de calculatoare. Din contră, *protocoalele* OSI nu au devenit populare. Pentru TCP/IP este adevărată afirmaţia inversă: *modelul* este practic inexistent, dar *protocoalele* sunt larg utilizate. Dat fiind faptul că informaticienilor le place să prepare – şi apoi să şi mănânce – propria lor prăjitură, în această carte vom folosi un model OSI modificat, dar ne vom concentra în primul rând pe TCP/IP şi alte protocoale înrudite cu el; de asemenea, vom folosi şi protocoale mai noi, precum 802, SONET şi Bluetooth. Modelul de lucru folosit în carte este modelul hibrid prezentat în fig. l-24.

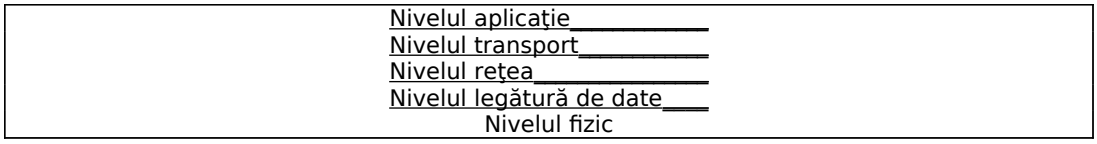


Fig. l-24. Modelul hibrid de referinţă care va fi utilizat în această carte.

1.5 EXEMPLE DE REŢELE

Subiectul reţelelor de calculatoare acoperă diferite tipuri de reţele, mari şi mici, arhicunoscute sau mai puţin cunoscute. Ele au scopuri, dimensiuni şi tehnologii diverse. În următoarele secţiuni, vom studia câteva exemple, pentru a avea o idee despre varietatea pe care o poate regăsi oricine în domeniul reţelelor de calculatoare.

Vom porni cu Internet-ul, probabil cea mai cunoscută reţea, şi vom studia istoria, evoluţia şi tehnologiile sale. Apoi vom discuta ATM, care este de multe ori utilizată în nucleul reţelelor (telefonice) mari. Din punct de vedere tehnic, este destul de diferită de Internet, ceea ce evidenţiază un contrast interesant. Apoi vom introduce Ethernet, dominantă în cazul reţelelor locale. În final, vom studia IEEE 802.11, standardul pentru reţele fără cablu.

1.5.1 Internet

Internet-ul nu este deloc o reţea, ci o colecţie vastă de reţele diverse, care utilizează anumite protocoale comune şi oferă anumite servicii comune. Este un sistem neobişnuit prin aceea că nu a

fost planificat de nimeni şi nu este controlat de nimeni. Pentru a-l înţelege mai bine, să pornim de la începuturi şi să vedem cum s-a dezvoltat şi de ce. Pentru o istorie foarte reuşită a Internet-ului, este recomandată cartea lui John Naughton (2000). Este una dintre acele cărţi rare care nu este numai plăcută la citit, dar are şi 20 de pagini de ibid. şi op.cit. pentru istoricii serioşi. Unele dintre materialele de mai jos sunt bazate pe această carte.

Desigur, au fost scrise nenumărate cărţi tehnice despre Internet, şi despre protocoalele sale de asemenea. Pentru mai multe informaţii vedeţi, de exemplu, (Maufer, 1999).

ARPANET-ul

Povestea începe la sfârşitul anilor 1950. În momentul în care Războiul Rece era la apogeu, DoD (Department of Defense, rom: Departamentul de Apărare al SUA) a vrut o reţea de comandă şi control care să poată supravieţui unui război nuclear. La momentul acela, toate comunicaţiile militare foloseau reţelele telefonice publice, care erau considerate vulnerabile. Motivul pentru o astfel de părere poate fi observat în fig. l-25(a). Aici punctele negre reprezintă oficii de comutare, la ele fiind conectate mii de telefoane. Aceste oficii erau, la rândul lor, conectate la oficii de comutare de nivel mai înalt (oficii de taxare), pentru a forma o ierarhie naţională cu un nivel scăzut de redundanţă. Vulnerabilitatea sistemului consta în aceea că distrugerea câtorva oficii de taxare putea fragmenta sistemul în mai multe insule izolate.

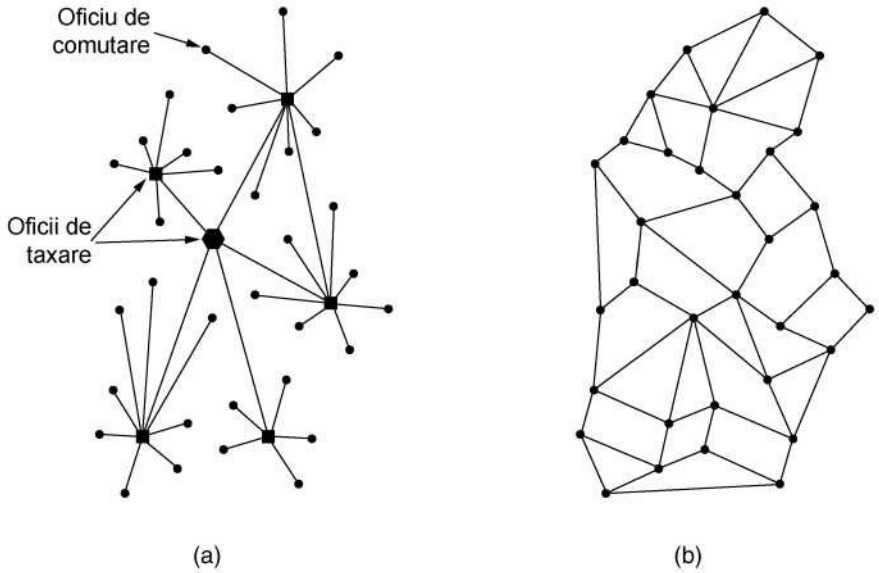


Fig. l-25. (a) Structura sistemului de telefonie. (b) Sistemul distribuit cu comutare al lui Baran.

În jurul anului 1960, DoD a oferit un contract corporaţiei RAND pentru a găsi o soluţie. Unul dintre angajaţii ei, Paul Baran, a venit cu ideea sistemului distribuit cu un nivel ridicat de toleranţa la defecte, prezentat în fig. l-25(b). Deoarece căile

dintre oricare două oficii de comutare erau în acest caz mult mai lungi decât căile pe care semnale analogice puteau să circule fără distorsiuni, Baran a propus utilizarea unei tehnologii digitale cu comutare de pachete prin întregul sistem. Baran a scris câteva rapoarte pentru DoD în care a descris ideile sale în detaliu. Oficialii de la Pentagon au agreat conceptul şi au apelat la AT&T, apoi la monopolul naţional al telefoniei SUA pentru a construi un prototip. AT&T a desconsiderat imediat ideile lui Baran. Cea mai mare şi cea mai bogata companie din lume nu avea de gând să permită unui tânăr oarecare să spună cum să se construiască un sistem de telefonie. Ei au declarat că sistemul propus de Baran nu poate fi construit, şi ideea a fost abandonată.

Au mai trecut câţiva ani şi DoD încă nu avea un sistem de comandă şi control mai bun. Pentru a înţelege ceea ce s-a întâmplat în continuare trebuie să ne întoarcem în Octombrie 1957, când Uniunea Sovietică a întrecut SUA în domeniul spaţial prin lansarea primului satelit artificial, Sputnik. Când preşedintele Eisenhower a încercat să afle cine adormise la comandă, a fost surprins să afle că Armata, Marina şi Forţele Aeriene îşi disputau bugetul de cercetare al Pentagonului. Răspunsul lui imediat a fost crearea unei singure organizaţii de cercetare în domeniul apărării: ARPA (Advanced Research Projects Agency, rom: Agenţia de Cercetare pentru Proiecte Avansate). ARPA nu avea nici oameni de ştiinţă, nici laboratoare; de fapt, nu avea decât un birou şi un mic buget (după standardele Pentagonului). Îşi ducea misiunile la îndeplinire prin acordarea de granturi (fonduri pentru cercetare) şi contracte universităţilor şi companiilor ale căror idei păreau promiţătoare.

În primii câţiva ani, ARPA a încercat să afle care îi era misiunea. În 1967, atenţia directorului Larry Roberts a fost atrasă de domeniul reţelelor. A contactat diverşi experţi ca să decidă ce este de făcut. Unul dintre ei, Weslez Clark, a sugerat construirea unei subreţele cu comutare de pachete, dând fiecărei gazde propriul ruter, aşa cum este ilustrat în fig. l-12.

După un oarecare scepticism iniţial, Roberts a adoptat ideea şi a prezentat o lucrare destul de vagă despre ea la Simpozionul ACM SIGOPS ţinut în Gatlinburg, Tennessee la sfârşitul lui 1967 (Roberts, 1967). Spre surprinderea lui Roberts, o altă lucrare prezentată la aceeaşi conferinţă descria un sistem similar, care nu numai că fusese proiectat, dar fusese şi implementat sub comanda lui Donald Davies de la NPL (National Physical Laboratories, rom: Laboratoarele Naţionale de cercetări în Fizică), Anglia. Sistemul propus de NPL nu era un sistem naţional (conecta numai câteva calculatoare în campusul NPL) dar demonstrase că comutarea de pachete poate fi funcţională. În plus, cita din rapoartele timpurii ale lui Baran care fuseseră desconsiderate la momentul respectiv. Roberts s-a întors de la Gatlinburg hotărât să construiască ceva ce urma să devină cunoscut sub numele de ARPANET.

Subreţeaua trebuia să fie formată din minicalculatoare numite **IMP**-uri (**Interface Message Processors –** procesoare de mesaje de interfaţă) conectate prin linii de transmisie. Pentru o siguranţă mare, fiecare IMP trebuia legat la cel puţin alte două IMP-uri. Subreţeaua avea să fie o subreţea datagramă, astfel că dacă unele linii şi IMP-uri se defectau, mesajele puteau fi redirijate automat pe căi

alternative.

Fiecare nod al reţelei era format dintr-un IMP şi dintr-o gazdă, aflate în aceeaşi încăpere şi legate printr-un fir scurt. O gazdă putea să trimită mesaje de până la 8063 biţi spre IMP-ul său, iar acesta descompunea apoi mesajele în pachete de cel mult 1008 biţi şi le retransmitea la destinaţie separat. Fiecare pachet era primit în întregime înainte de a fi reexpediat, astfel că subreţeaua a fost prima reţea electronică memorează-şi-retransmite cu comutare de pachete.

ARPA a căutat apoi o ofertă pentru construirea subreţelei. Au depus oferte douăsprezece firme. După evaluarea tuturor propunerilor, ARPA a selectat BBN, o firmă de consultanţă din Cambridge, Massachusetts, şi în 1968 a încheiat cu aceasta un contract pentru construirea subreţelei şi scrierea programelor de subreţea. BBN a decis să utilizeze pe post de IMP-uri minicalculatoare Honeywell DDP-316 special modificate, dispunând de o memorie internă de 12K cu cuvinte pe 16 biţi. IMP- urile nu aveau discuri, pentru că părţile mobile erau considerate nesigure. IMP-urile au fost interconectate prin linii de 56 Kbps închiriate de la companii de telefoane. Deşi 56 Kbps este acum o variantă pentru adolescenţii care nu îşi permit ADSL sau cablu, era cea mai bună alternativă a momentului respectiv.

Programele au fost împărţite în două: pentru subreţea şi pentru gazde. Programele de subreţea cuprind gestionarea capătului dinspre IMP al conexiunii gazdă-IMP, protocolul IMP-IMP şi un protocol sursă IMP – destinaţie IMP, proiectat pentru a mări siguranţa. Proiectul iniţial al reţelei ARPANET este prezentat în fig. l-26.

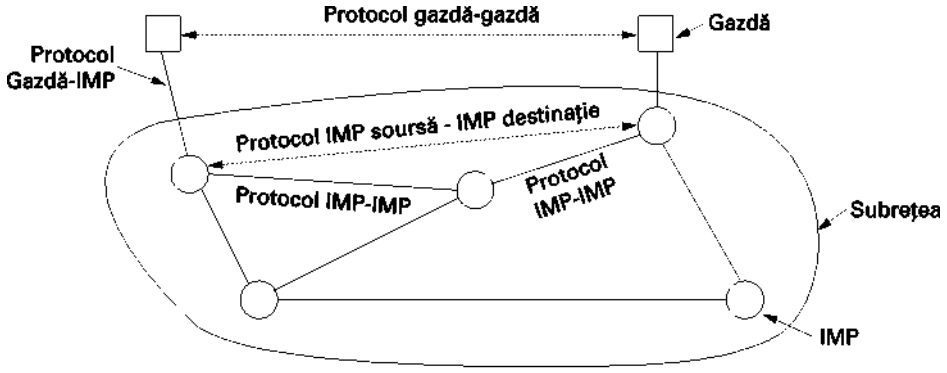


Fig. 1-26. Proiectul iniţial al reţelei ARPANET.

Şi în afara subreţelei erau necesare programe: gestionarea capătului dinspre gazdă al conexiunii gazdă-IMP, protocolul gazdă-gazdă şi programe de aplicaţie. În scurt timp, a devenit clar că BBN considera sarcina sa încheiată din momentul în care acceptase un mesaj pe un fir gazdă-IMP şi îl plasase pe firul gazdă-IMP destinaţie.

Robets avea o nouă problemă: gazdele aveau şi ele nevoie de programe. Pentru a rezolva aceasta problemă, el a convocat o adunare a cercetătorilor în reţele, majoritatea fiind tineri absolvenţi de facultate, la Snowbird, în Utah, în vara anului 1969. Absolvenţii se aşteptau ca nişte experţi în reţele să le explice proiectarea şi

software-ul reţelei şi ca fiecare din ei să primească după aceea sarcina de a scrie o parte din programe. Au rămas însă muţi de uimire când au constatat că nu exista nici un expert în reţele şi nici o proiectare serioasă. Trebuiau să îşi dea seama singuri ce au de făcut.

Cu toate acestea, în decembrie 1969 începea deja să funcţioneze o reţea experimentală cu patru noduri, la UCLA, UCSB, SRI şi Universitatea din Utah. Au fost alese aceste patru instituţii pentru că toate aveau un număr mare de contracte cu ARPA şi toate aveau calculatoare gazdă diferite şi complet incompatibile (doar ca treaba să fie mai amuzantă). Pe măsură ce se aduceau şi se instalau mai multe IMP-uri, reţeaua creştea rapid; în scurt timp, s-a întins pe tot spaţiul Statelor Unite. Fig. l-27 arată cât de repede a crescut ARPA în primii 3 ani.

Pe lângă ajutorul oferit pentru dezvoltarea tânărului ARPANET, ARPA a finanţat de asemenea cercetări în domeniul reţelelor de sateliţi şi reţelelor mobile radio cu pachete. Într-o faimoasă demonstraţie, un camion care circula în California folosea reţeaua radio cu pachete pentru a trimite mesaje către SRI, aceste mesaje erau retransmise apoi prin ARPANET pe Coasta de Est, iar de aici mesajele erau expediate către University College din Londra prin reţeaua de sateliţi. Acest lucru permitea unui cercetător din camion să utilizeze un calculator din Londra în timp ce călătorea prin California.

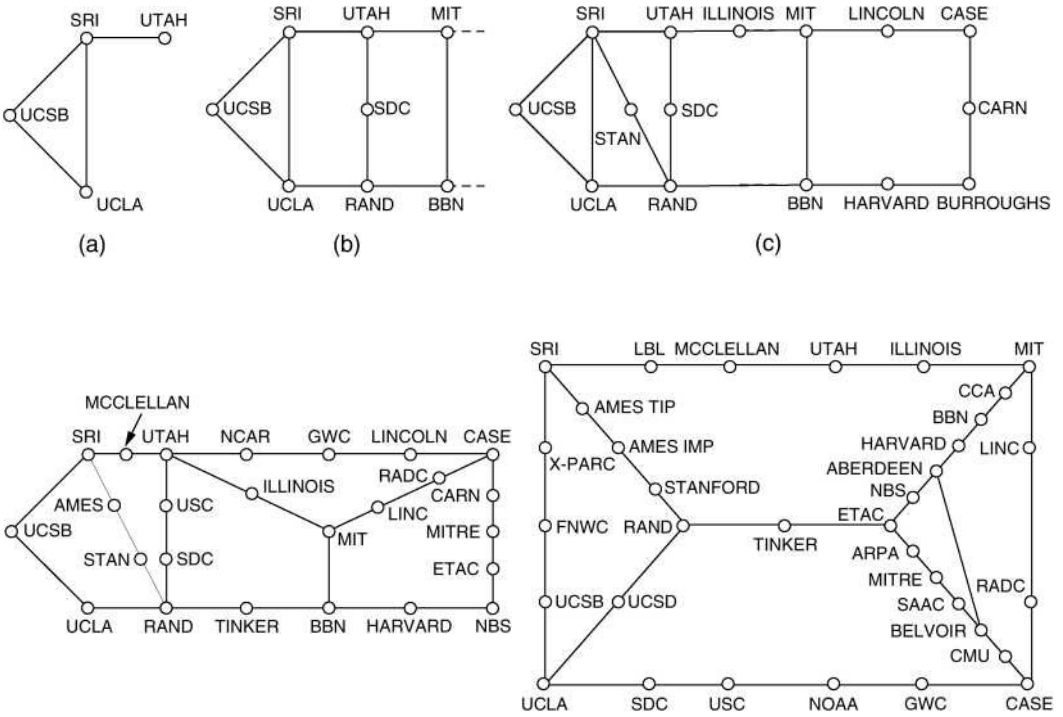


Fig. l-27. (a) Dec.. 1969. (b) Iulie 1970. (c) Martie 1971. (d) Aprilie 1972.

Acest experiment a demonstrat totodată că protocoalele ARPANET existente nu erau potrivite pentru a rula pe mai multe reţele. Observaţia a condus la noi cercetări asupra protocoalelor, culminând cu invenţia modelului şi protocoalelor TCP/IP (Cerf şi Kahn, 1974). TCP/IP a fost proiectat special pentru a trata comunicarea prin inter-reţele, un lucru care devenea din ce în ce mai important, pe măsură ce tot mai multe reţele erau legate la ARPANET.

Pentru a încuraja adoptarea acestor noi protocoale, ARPA a semnat câteva contracte cu BBN şi cu University of California din Berkeley pentru a integra protocoalele în Berkeley UNIX. Cercetătorii de la Berkeley au dezvoltat o interfaţă de programare a reţelei (soclurile) şi au scris numeroase aplicaţii, utilitare şi programe de administrare care să simplifice interconectarea.

Momentul era ideal. Multe universităţi tocmai achiziţionaseră un al doilea sau al treilea calculator VAX şi un LAN care să le conecteze, dar nu aveau nici un fel de programe de interconectare. Când a apărut 4.2BSD, cu TCP/IP, socluri şi multe utilitare de reţea, pachetul complet a fost adoptat imediat. Mai mult chiar, folosind TCP/IP, LAN-urile se puteau lega simplu la ARPANET şi multe LAN-uri au făcut acest lucru.

În anii ‘80 au fost conectate la ARPANET multe alte reţele, în special LAN-uri. Pe măsură ce creştea dimensiunea reţelei, găsirea gazdelor devenea tot mai costisitoare; de aceea, a fost creat **DNS** (**Domain Name System**, rom: Sistemul Numelor de Domenii), care organiza maşinile în domenii şi punea în corespondenţă

numele gazdelor cu adrese IP. De atunci încoace, DNS a ajuns să fie un sistem de baze de date distribuit, generalizat, folosit pentru a memora diverse informaţii referitoare la procedurile de atribuire a numelor. Vom studia detaliat acest sistem în cap. 7.

NSFNET

La sfârşitul anilor 1970, NSF (U.S. National Science Foundation, rom: Fundaţia Naţională de Ştiinţă din SUA) a remarcat impactul imens pe care ARPANET-ul îl avea asupra cercetării universitare, reţeaua permiţând savanţilor din toată ţara să partajeze date şi să colaboreze la proiecte de cercetare. Dar, pentru a se conecta la ARPANET, o universitate trebuia să aibă un contract de cercetare cu DoD, iar multe universităţi nu aveau. Răspunsul NSF a fost proiectarea unui succesor al ARPANET care să fie deschis tuturor grupurilor de cercetare din universităţi. Pentru a avea ceva concret de la care să pornească, NSF a decis să construiască o reţea tip coloană vertebrală (backbone) pentru a conecta cele 6 centre de supercalculatoare pe care le deţinea în San Diego, Boulder, Champaign, Pittsburgh, Ithaca, Princeton. Fiecărui calculator i s-a dat un frate mai mic, care era de fapt un microcalculator LSI-11 denumit **ftizzball**. Aceste fuzzball-uri erau conectate cu linii închiriate de 56 Kbps şi formau o subreţea , care folosea aceeaşi tehnologie ca şi ARPANET. Tehnologia programelor era însă diferită: fuzzball-urile au fost proiectate pentru a conversa direct folosind TCP/IP, ceea ce a condus la crearea primei reţele pe arie larga bazată pe TCP/IP (TCP/IP WAN).

NSF a finanţat, de asemenea, un număr de (aproximativ 20, până la urmă) reţele regionale care se conectau la coloana vertebrală, permiţând utilizatorilor din mii de universităţi, laboratoare de cercetare, biblioteci şi muzee să acceseze oricare dintre supercalculatoare şi să comunice între ei. Reţeaua completă, care includea coloana vertebrală şi reţelele regionale, a fost numită **NSFNET**. Aceasta a fost conectată la ARPANET printr-o legătură între un IMP şi un fuzzball din laboratorul de la Carnegie-Mellon. Prima coloană vertebrală NSFNET este ilustrată în fig. l-28.

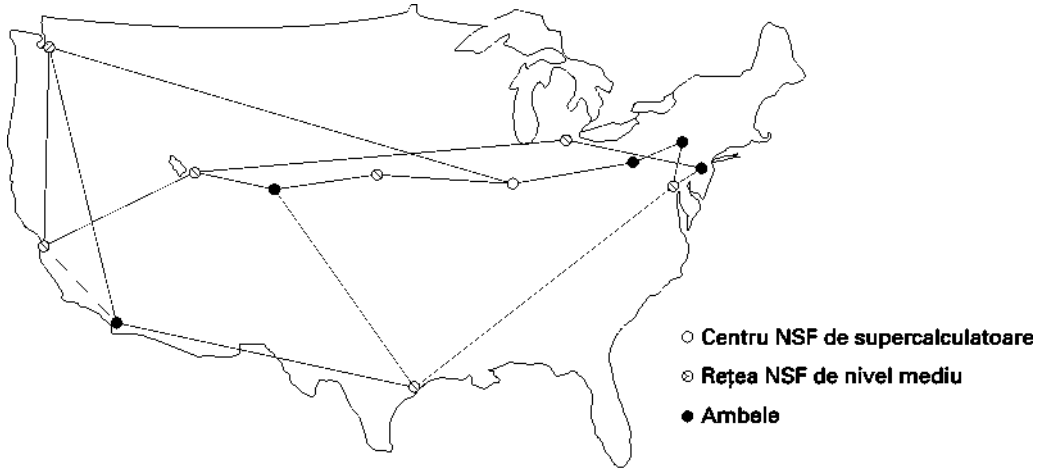


Fig. 1-28. Coloana vertebrală NSFNET în 1988.

NSFNET-ul a reprezentat un succes foarte rapid şi a fost suprasolicitat din clipa în care a început să funcţioneze. NSF a început imediat să planifice succesorul NSFNET-ului şi a semnat un contract cu consorţiul MERIT cu sediul în Michigan. Pentru realizarea coloanei vertebrale numărul 2, au fost închiriate de la MCI (care a fuzionat între timp cu WorldCom) canale cu fibre optice de 448 Kbps. Ca rutere s-au folosit IBM PC-RT. Şi această reţea a devenit curând supraîncărcată, drept care, în 1990, a doua coloană vertebrală a fost adusă la viteza de 1.5 Mbps.

Dar creşterea a continuat, iar NSF a realizat că guvernul nu poate finanţa interconectările la nesfârşit. În plus, o serie de organizaţii comerciale erau interesate să se conecteze, dar statutul NSF le interzicea să se lege la reţele finanţate de NSF. În consecinţă, NSF a încurajat MERIT, MCI şi IBM să formeze o corporaţie nonprofit, **ANS** (**Advanced Networks and Services**, rom: reţele şi servicii avansate), ca un pas pe drumul spre comercializare. În 1990, ANS a preluat NSFNET şi a înlocuit legăturile de 1.5 Mbps cu legături de 45 Mbps, formând **ANSNET**. Această reţea a funcţionat timp de 5 ani şi apoi a fost cumpărată de America Online. Dar până atunci, diverse companii ofereau deja servicii IP comerciale şi era clar că guvernul trebuia să se retragă din afacerea cu reţele.

Ca să uşureze tranziţia şi ca să fie sigur că orice reţea regională putea comunica cu orice altă reţea regională, NSF a semnat contracte cu patru operatori de reţele diferiţi în vederea stabilirii unui **NAP** (**Network Access Point**, rom: punct de acces la reţea). Aceşti operatori erau PacBell (San Francisco), Ameritech (Chicago), MFS (Washington, D.C.), şi Sprint (New York City, unde – din raţiuni legate de NAP – Pennsauken, N.J. se consideră New York City). Fiecare operator de reţea care dorea să ofere servicii de infrastructură pentru reţelele regionale NSF trebuia să se lege la toate NAP-urile.

De aceea, pentru a ajunge de la NAP-ul său la NAP-ul destinaţiei, un pachet trimis din orice reţea regională putea opta între mai multe companii care oferă servicii de transmisie pe coloana vertebrală. În consecinţă, pentru a fi alese de reţelele regionale, companiile de comunicaţii au fost forţate să intre în competiţie pe baza serviciilor şi preţurilor practicate – bineînţeles, aceasta era ideea. Ca rezultat, conceptul unei singure reţele de tip coloană vertebrală a fost înlocuit de o infrastructură competitivă condusă de criterii comerciale. Multora le place să critice Guvernul Federal pentru că nu este destul de inovator, dar în zona reţelelor, DoD şi NSF au fost cele care au creat infrastructura care a stat la bazele formării Internet-ului şi apoi a cedat-o industriei pentru operare şi exploatare.

În timpul anilor 1990, multe alte ţări şi regiuni construiesc şi ele reţele naţionale, de multe ori modelate chiar după ARPANET şi NSFNET. Acestea includ EuropaNET şi EBONE în Europa, care au pornit cu linii de 2 Mbps şi apoi au avansat până la linii de 34 Mbps. În cele din urmă, şi infrastructura de reţea din Europa a fost cedată industriei spre operare şi exploatare.

Folosirea Internet-ului

Numărul reţelelor , maşinilor şi utilizatorilor conectaţi la ARPANET a crescut rapid după ce TCP/IP a devenit, la 1 ian. 1983, unicul protocol oficial. Când au fost conectate NSFNET şi ARPANET, creşterea a devenit exponenţială. S-au alăturat

multe reţele regionale şi s-au realizat legături cu reţele din Canada, Europa şi Pacific.

Cândva, pe la mijlocul anilor 1980, lumea a început să vadă colecţia de reţele ca fiind un internet, iar apoi ca fiind Internet-ul; nu a existat însă nici un toast oficial cu politicieni desfăcând sticle de şampanie.

Substanţa care ţine legat Internet-ul este modelul de referinţă TCP/IP şi stiva de protocoale TCP/IP. TCP/IP face posibile serviciile universale, putând fi comparată cu adoptarea lăţimii standard pentru căile ferate în secolul 19 sau cu adoptarea protocoalelor comune de semnalizare de către toate companiile telefonice.

Ce înseamnă de fapt să fii pe Internet? Definiţia noastră este că o maşină este pe Internet dacă foloseşte stiva de protocoale TCP/IP, are o adresă IP şi are posibilitatea de a trimite pachete IP către toate celelalte maşini de pe Internet. Simpla posibilitate de a trimite şi primi poştă electronică nu este suficientă, deoarece poşta electronică este redirectată către multe reţele din afara Internet-ului. Oricum, subiectul este cumva umbrit de faptul că milioane de calculatoare personale pot să apeleze un furnizor de servicii Internet folosind un modem, să primească o adresă IP temporară şi apoi să trimită pachete IP spre alte gazde. Are sens să privim asemenea maşini ca fiind pe Internet numai atâta timp cât ele sunt conectate la ruterul furnizorului de servicii.

Tradiţional (însemnând din 1970 până în jurul lui 1990), Internet-ul şi predecesorii săi au avut patru aplicaţii principale, după cum urmează:

1. **Poşta electronică**. Facilitatea de a compune, trimite şi primi poştă electronică a existat din primele zile ale ARPANET-ului şi este extrem de populară. Mulţi oameni primesc zeci de mesaje pe zi şi consideră poşta electronică principalul lor mijloc de a interacţiona cu lumea exterioară, depăşind de departe telefonul şi poşta obişnuită. Programele de poştă electronică sunt astăzi disponibile practic pe orice tip de calculator.
2. **Ştiri**. Grupurile de ştiri sunt forumuri specializate în care utilizatorii cu un anumit interes comun pot să facă schimb de mesaje. Există mii de grupuri de ştiri, pe subiecte tehnice sau non-tehnice incluzând calculatoarele, ştiinţa, divertismentul şi politica. Fiecare grup de ştiri are eticheta, stilul şi obiceiurile sale proprii şi nenorocirile se vor abate asupra celor care le încalcă.
3. **Conectare la distanţă**. Folosind programe ca telnet, rlogin sau ssh, utilizatorii aflaţi oriunde pe Internet pot să se conecteze la orice maşină pe care au un cont.
4. **Transfer de fişiere**. Copierea fişierelor de pe o maşină din Internet pe alta este posibilă utilizând programul FTP. În acest fel sunt disponibile extrem de multe articole, baze de date şi alte informaţii.

Până la începutul anilor 1990 Internet-ul a fost foarte populat cu cercetători din domeniul academic, guvernamental şi industrial. O aplicaţie nouă, **WWW** (**World Wide Web**), a schimbat total situaţia şi a adus în reţea milioane de noi utilizatori care nu fac parte din mediul academic. Această aplicaţie, inventată de fizicianul Tim Berners Lee de la CERN, nu a modificat nici una din facilităţile existente, în

schimb le-a făcut mai uşor de folosit. Împreună cu programul de navigare Mosaic, scris la Centrul Naţional pentru Aplicaţiile Supercalculatoarelor, WWW-ul a făcut posibil ca un sit să pună la dispoziţie un număr de pagini de informaţii conţinând text, poze, sunet şi chiar video, în fiecare pagină existând legături către alte pagini. Printr-un clic pe o legătură, utilizatorul este imediat transportat la pagina indicată de legătură. De exemplu, multe firme au o pagină principală cu intrări care trimit la alte pagini pentru informaţii asupra produselor, liste de preţuri, reduceri, suport tehnic, comunicare cu angajaţii, informaţii despre acţionari şi multe altele.

Într-un timp foarte scurt au apărut numeroase alte tipuri de pagini: hărţi, tabele cu cotaţii la bursă, cataloage de bibliotecă, programe radio înregistrate şi chiar o pagină care oferă legături spre textele complete ale multor cărţi cărora le-au expirat drepturile de autor (Mark Twain, Charles Dickens, etc.). De asemenea, mulţi oameni au pagini personale (home pages).

Mare parte din creşterea Internetului în timpul anilor 1990 a fost alimentată de companii denumite **ISP** (Internet Service Providers, rom: Furnizori de Servicii Internet). Acestea sunt companii care oferă utilizatorilor individuali posibilitatea de a apela, de acasă, una dintre maşinile furnizorului şi de a se conecta la Internet, obţinând în consecinţă acces la poşta electronica, WWW şi alte servicii similare. La sfârşitul anilor 1990, aceste companii au înregistrat zeci de milioane de noi utilizatori în fiecare an, modificând astfel complet caracterul reţelei, care s-a transformat dintr-o reţea academică şi militară într-o utilitate publică, precum sistemul de telefonie. Numărul actual al utilizatorilor Internet nu este cunoscut, dar este cu siguranţă de ordinul sutelor de milioane la nivel mondial şi probabil că va ajunge la un miliard în curând.

Arhitectura Internet

În această secţiune vom încerca să aruncăm o scurtă privire de ansamblu asupra Internet-ului de astăzi. Din cauza multor fuziuni între companiile de telefoane şi companiile ISP, apele au devenit tulburi. şi este de cele mai multe ori dificil de precizat care sunt atribuţiile fiecăruia, cine ce anume are de făcut. În consecinţă această descriere va fi simplificată în raport cu realitatea efectivă. Imaginea de ansamblu este prezentată în fig. l-29. În continuare, vom analiza această figură bucată cu bucată.

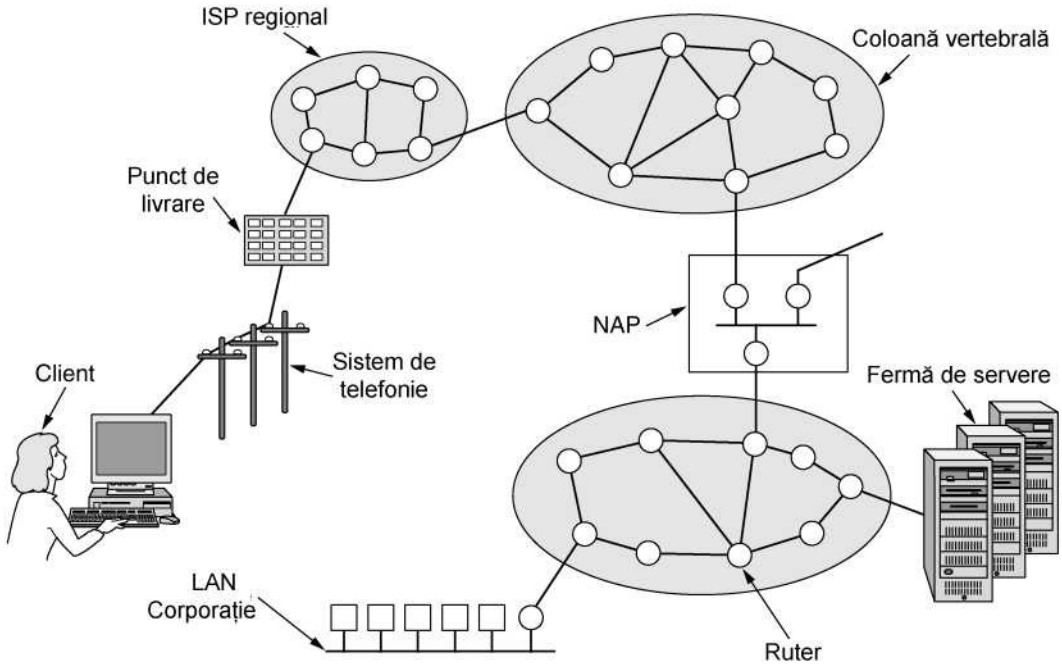


Fig. l-29. Privire de ansamblu asupra Internet-ului.

Un bun punct de pornire este sistemul propriu al clientului. S ă presupunem că acest client sună la ISP-ul său printr-o linie telefonică, aşa cum se vede în fig. l-29. Modemul este o placă din PC-ul clientului care converteşte semnalele digitale pe care le produce calculatorul în semnale analogice care pot circula fără probleme prin sistemul telefonic. Aceste semnale sunt transferate la punctul de livrare (POP) al ISP-ului, unde sunt preluate din sistemul telefonic şi injectate în reţeaua regionala a ISP. De aici înainte, sistemul este în întregime digital şi foloseşte comutarea de pachete. Daca ISP-ul este acelaşi cu furnizorul local de telefonie, punctul de livrare va fi localizat, probabil, chiar în oficiul de comutare al serviciului telefonic, punctul în care se termină firul de telefon al utilizatorului. Chiar dacă ISP-ul nu este acelaşi cu furnizorul local de telefonie, punctul de livrare poate fi doar la distanţă de câteva oficii de comutare.

Reţeaua regională a ISP este formată prin interconectarea ruterelor din diverse oraşe pe care le deserveşte compania. Dacă pachetul este destinat unei gazde deservite direct de către reţeaua ISP, pachetul va fi livrat direct gazdei. Altfel, el

este livrat în continuare operatorului care furnizează companiei ISP servicii de comunicare prin coloana vertebrală (backbone) a reţelei.

În partea superioară a acestei ierarhii sunt operatorii principali de la nivelul de coloană vertebrală a reţelei, companii precum AT&T sau Sprint. Aceştia operează coloane vertebrale mari, internaţionale, cu mii de rutere conectate prin fibra optică cu bandă largă de transfer. Corporaţiile mari şi firmele care oferă servicii de găzduire (hosting), utilizează ferme de servere (maşini care pot servi mii de pagini Web pe secundă) sunt conectate adeseori direct la nivelul coloanei vertebrale. Operatorii încurajează această conectare directă prin închirierea de spaţii în ceea ce se numeşte **„hotelul companiei de transport”** (**carrier hotel**), şi reprezintă de cele mai multe ori **sertare (racks)** pentru echipamente aflate în aceeaşi cameră cu ruterul, pentru a permite conexiuni scurte şi rapide între fermele de servere şi coloana vertebrală a reţelei.

Dacă un pachet trimis în coloana vertebrală este destinat unui ISP sau unei companii deservite de aceeaşi coloană, el este transmis celui mai apropiat ruter. Oricum există multe astfel de coloane vertebrale în întreaga lume, astfel încât un pachet poate să treacă într-o coloană concurentă. Pentru a permite pachetelor să treacă dintr-o coloană în alta, toate aceste coloane principale sunt conectate în NAP-urile (Network Access Point, rom: Punct de acces în reţea) discutate mai devreme. În principiu, un NAP este o cameră plină cu rutere, cel puţin unul pentru fiecare coloană vertebrală conectată. O reţea locală camerei conectează toate aceste rutere, astfel încât pachetele să poată fi retransmise din orice coloană în orice altă coloană. În afară de interconectarea în NAP-uri, coloanele vertebrale de dimensiuni mari au numeroase conexiuni directe între ruterele lor, tehnică denumită **conectare privată** (**private peering**). Unul dintre multiplele paradoxuri ale Internet-ului este acela că ISP-urile care sunt la nivel public în competiţie pentru clienţi, cooperează de cele mai multe ori pentru a realiza astfel de conectări private (private peering) (Metz, 2001).

Astfel se încheie acest scurt tur de orizont asupra Internet-ului. Vom avea multe de spus despre componentele individuale şi proiectarea lor, despre algoritmi şi despre protocoale în capitolele următoare. Merită de asemenea menţionat în trecere că anumite companii şi-au interconectat toate reţelele interne existente, folosind de multe ori aceleaşi tehnologii ca şi Internet-ul. Aceste **intraneturi** sunt accesibile de cele mai multe ori numai din interiorul companiei, dar altfel funcţionează la fel ca Internet-ul.

1.5.5 Reţele orientate pe conexiune

Încă de la începuturile domeniului reţelelor, există un război între cei care susţin subreţelele fără conectare (de exemplu datagramele) şi cei care susţin subreţelele orientate pe conexiune. Susţinătorii subreţelelor fără conexiune provin din comunitatea ARPANET/Internet. Amintiţi-vă că dorinţa iniţială a DoD în finanţarea şi construirea ARPANET a fost să aibă o reţea care să continue să funcţioneze chiar şi după ce mai multe lovituri nucleare îndreptate direct împotriva ei au distrus numeroase rutere şi linii de transmisie. De aceea, toleranţa la defecte se afla pe

primele poziţii ale listei de priorităţi; taxarea clienţilor nu exista pe acea listă. Această abordare a condus la o proiectare fără conexiune în care fiecare pachet era rutat independent de orice alt pachet. Ca o consecinţă, dacă anumite rutere se defectează în timpul unei sesiuni, nu apare nici o problemă atâta timp cât sistemul se poate reconfigura singur, dinamic, astfel încât pachetele următoare să găsească o rută către destinaţie, chiar dacă ea este diferită de cea utilizată până la momentul respectiv.

Tabăra celor care susţin reţelele orientate conexiune provine din lumea comunicaţiilor pe linii telefonice. În sistemul telefonic, un utilizator trebuie să formeze numărul pe care doreşte să îl apeleze şi să aştepte formarea unei conexiuni înainte de a vorbi sau de a transmite date. Aceasta fază de conectare stabileşte o rută prin sistemul telefonic, rută care va fi menţinută până când apelul este încheiat. Toate cuvintele sau pachetele de date urmează aceeaşi rută. Dacă o linie sau un comutator de pe respectiva cale se defectează, apelul este încheiat forţat. Aceasta proprietate era exact cea care nu convenea deloc Departamentului de Apărare.

De ce sunt companiile organizate astfel? Din două motive:

1. Calitatea serviciilor
2. Facturarea

Prin setarea unei conexiuni în avans, subreţeaua poate rezerva resurse precum zone tampon de memorie sau capacitatea de procesare a procesorului din ruter. Dacă se face o încercare de a iniţia un apel şi nu se găsesc suficiente resurse disponibile, apelul este rejectat şi apelantul primeşte un fel de semnal de „ocupat”. În acest fel, de îndată ce conexiunea a fost stabilită, conexiunea va obţine servicii bune din punct de vedere calitativ. Într-o reţea fără conexiune, dacă prea multe pachete ajung la acelaşi ruter în acelaşi moment, ruterul va fi sufocat şi, probabil, va pierde din pachete. Eventual, utilizatorul va observa şi le va retrimite, dar calitatea serviciilor va fi proastă şi deloc potrivită pentru comunicaţii audio sau video, cu excepţia cazurilor în care reţeaua este doar foarte puţin încărcată. Nu mai este nevoie să precizăm că pentru companii calitatea de transmitere a semnalului audio este un parametru extrem de important, şi de aceea preferă reţelele orientate pe conexiune.

Cel de-al doilea motiv pentru care companiile de telefonie preferă serviciile orientate pe conexiune este acela că sunt obişnuite să taxeze utilizatorul în funcţie de timpul de conexiune. Atunci când se face un apel la distanţă (chiar şi local, dar în afara Americii de Nord) taxarea se face la minut. La apariţia reţelelor, aceste companii au fost automat atrase în acest sistem, în care taxarea la minut era uşor de făcut. Dacă trebuie stabilită o conexiune înainte de transmisia propriu-zisă a datelor, ceasul de taxare este pornit. Dacă nu există conexiune, nu poţi fi taxat pentru ea.

Culmea, menţinerea sistemului de taxare este foarte scumpă. Dacă o companie de telefonie ar trebui să adopte o schemă de plată cu rate lunare fixe, fără a ţine cont de numărul de apeluri şi fără a ţine evidenţa facturărilor pe convorbire, cu siguranţă s-ar economisi sume mari de bani, în ciuda creşterii însemnate a

numărului de apeluri care va rezulta. Factorii politici, de reglementare şi de altă natură sunt însă împotrivă. Destul de interesant este că o astfel de politică este funcţională în alte sectoare. De exemplu, cablul TV este facturat cu o rată lunară fixă, indiferent de cât de mult te uiţi la televizor. Ar fi putut să fie proiectat şi având la bază un principiu plată-pentru-utilizare (pay- per-view), dar nu s-a făcut aşa, în parte şi din cauza cheltuielilor impuse de o asemenea strategie de facturare (dată fiind calitatea slabă a majorităţii televiziunilor, trebuie luat în considerare chiar şi factorul „jenă"). Un alt exemplu sunt parcurile tematice care încasează o taxă de intrare zilnică, spre deosebire de caravane, care taxează plimbarea.

Acestea fiind spuse, nu va fi o surpriză că toate reţelele proiectate de industria de telefonie au avut subreţele orientate pe conexiune. Ceea ce este probabil surprinzător este că şi Internet-ul deviază în aceasta direcţie, pentru a oferi o calitate mai bună pentru serviciile audio şi video. Vom reveni la acest subiect în cap. 5. Dar, să examinăm în continuare câteva reţele orientate pe conexiune.

X.25 şi Frame Relay (releu de cadre)

Primul exemplu de reţea orientată conexiune este X.25, care a fost prima reţea publică de date. A fost dată în folosinţă în anii 1970, într-un moment în care serviciile telefonice erau un monopol peste tot, şi compania de telefonie din fiecare ţară se aştepta să existe şi o reţea de date unică în ţară – a lor. Pentru a folosi X.25, un calculator a stabilit mai întâi o conexiune cu calculatorul aflat la distanţă, adică a făcut un apel telefonic. Pentru această conexiune s-a alocat un număr de conexiune folosit apoi în transferul pachetelor de date (deoarece pot fi deschise mai multe conexiuni în acelaşi timp). Pachetele de date erau foarte simple, fiind formate dintr-un antet de 3 ... 128 de octeţi de date. În antet se regăsea un număr de conexiune de 12 biţi, un număr de secvenţă al pachetului, un număr de confirmare pozitivă (ACK) şi câţiva biţi oarecare. Reţelele X.25 au funcţionat aproape un deceniu cu un oarecare succes.

În anii 1980, reţelele X.25 au fost înlocuite pe scară largă cu un nou tip de reţea, denumit **Frame Relay** (Releu de Cadre). În esenţă, este vorba de o reţea orientată pe conexiune, fără control al erorilor şi fără control al fluxului de date. Deoarece era orientată pe conexiune, pachetele erau furnizate în ordine (dacă erau furnizate). Aceste caracteristici – distribuire de pachete în ordine, lipsa de control al erorilor, lipsa de control al fluxului au făcut ca Frame Relay să se asemene cu o reţea locală de dimensiuni mari. Aplicaţia cea mai importantă a fost interconectarea reţelelor locale aflate în diverse birouri ale companiilor. Deşi Frame Relay a avut parte de un succes modest, este folosit şi astăzi în anumite companii.

ATM (Asynchronous Transfer Mode)

Încă o reţea orientată pe conexiune – una mult mai importantă de această dată – este ATM (ATM Asynchronous Transfer Mode, rom: Mod de Transfer Asincron). Acest nume, oarecum ciudat, este justificat prin aceea că, în timp ce în reţelele telefonice majoritatea transmisiilor sunt sincrone (strâns legate de un semnal de ceas), în reţelele ATM transmisiile nu sunt sincrone.

ATM a fost proiectat la începutul anilor 1990 şi lansat la mijlocul acestei perioade incredibile (Ginsburg, 1996; Goralski, 1995; Ibe, 1997; Kim et al., 1994; at

Stallings, 2000). ATM urma să rezolve toate problemele de reţele şi telecomunicaţii ale lumii, unificând transmisiile de voce, date, televiziune prin cablu, telex, telegraf, porumbei mesageri, cutii de conserve conectate prin sfori, semnale cu fum, şi orice altceva într-un singur sistem integrat care să poată face totul pentru toată lumea. Nu s-a întâmplat. În mare parte, problemele erau similare cu acelea care au fost descrise mai devreme în ceea ce priveşte OSI, adică: ratarea momentului, tehnologii slabe, implementări ineficiente, politici proaste. După ce tocmai învinseseră companiile telefonice în runda I, mulţi membri din comunitatea Internet au văzut ATM-ul pe poziţia Internet-ului în lupta cu companiile mixte telefonie-ISP: Următorul. Dar nu a fost aşa, şi de această dată chiar şi cei mai fanatici susţinători ai datagramelor au trebuit să recunoască faptul că Internet-ul lăsa mult de dorit în privinţa calităţii serviciilor. Pentru a scurta povestea, ATM a înregistrat un succes mult mai mare decât OSI şi este acum utilizat pe scară largă în cadrul sistemelor de telefonie, adeseori vehiculând chiar pachete IP. Deoarece ATM este utilizat la ora actuală de majoritatea companiilor numai pentru operaţiile de rutare şi transport intern, în cele mai multe cazuri utilizatorii nu sunt conştienţi de existenţa lui, chiar dacă el este operaţional.

Circuite virtuale ATM

Deoarece reţelele ATM sunt orientate pe conexiune, transmisia datelor necesită mai întâi transmisia unui pachet pentru iniţializarea conexiunii. Pe măsură ce pachetul de iniţializare circulă prin subreţea, toate ruterele de pe drumul pe care îl parcurge îşi creează câte o înregistrare în tabelele de dirijare în care înregistrează existenţa conexiunii şi rezervă resursele necesare pentru ea. Conexiunile sunt de cele mai multe ori denumite circuite virtuale, în analogie cu circuitele fizice utilizate în sistemele de telefonie. Majoritatea reţelelor ATM suportă şi circuite virtuale permanente, care sunt conexiuni permanente între două gazde aflate la distanţă. Acestea sunt similare cu liniile închiriate din lumea telefoniei. Fiecare conexiune, fie ea temporară sau permanentă, are un identificator de conexiune unic. Un circuit virtual este prezentat în fig. l-30.

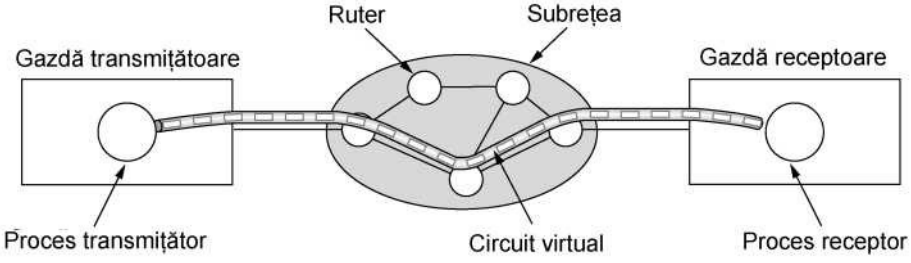


Fig. l-30. Un circuit virtual

Îndată ce o conexiune a fost stabilită, oricare dintre părţi poate să înceapă să transmită date. Ideea de bază în cazul reţelelor ATM este să se transmită toate informaţiile în pachete mici, de dimensiune fixă, denumite celule (cells). Celulele au 53 de octeţi, din care 5 octeţi reprezintă antetul, iar restul de 48 reprezintă

încărcătura efectivă, după cum se poate vedea în figura1-31. O parte din antet reprezintă identificatorul de conexiune, astfel încât atât transmiţătorul cât şi receptorul, precum şi toate ruterele intermediare pot şti corespondenţa dintre celule şi conexiuni (care celule aparţin cărei conexiuni). Această informaţie permite fiecărui ruter să dirijeze fiecare celulă pe care o primeşte. Dirijarea celulelor este implementată direct în partea hardware a ruterelor şi este o operaţie rapidă. De fapt, argumentul principal în alegerea de celule de dimensiune fixă este acela că este mai uşor de construit partea hardware pentru dirijare dacă ea are de a face cu pachete scurte şi egale ca dimensiune. Pachetele IP de lungime variabilă trebuie dirijate de programe (software), proces care este mai lent. Un alt avantaj al reţelelor ATM este acela că partea hardware poate fi configurată să multiplice o celul ă pe care o primeşte la intrare pe mai multe linii de ieşire, o proprietate obligatorie în cazul în care trebuie abordată transmisia unui program de televiziune difuzat către mai mulţi receptori. La urma urmei, celulele mici nu blochează nici o linie pentru prea mult timp, ceea ce face garantarea calităţii serviciilor mai uşoară.

Toate celulele urmează aceeaşi cale către destinaţie. Livrarea celulelor nu este garantată, dar ordinea lor da. Dacă doua celule 1 şi 2 sunt transmise în această ordine (1,2), dacă amândouă ajung, ele vor ajunge în aceeaşi ordine, niciodată nu va ajunge 2 înaintea lui 1. Dar oricare dintre ele, sau chiar amândouă se pot pierde pe drum. Este de datoria protocoalelor nivelului superior să repare eroarea cauzată de celulele pierdute. De reţinut că, deşi această garanţie nu este perfectă, este mai bună decât cea pe care o oferă Internet-ul. Acolo nu numai că pachetele se pot pierde, dar şi ordinea de ajungere la destinaţie poate fi oricare (nu are legătură cu ordinea de transmisie).

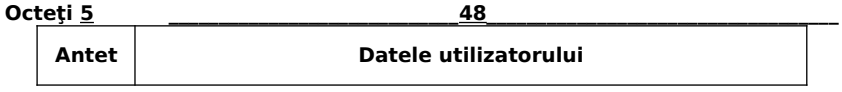


Fig. l-31. O celulă ATM

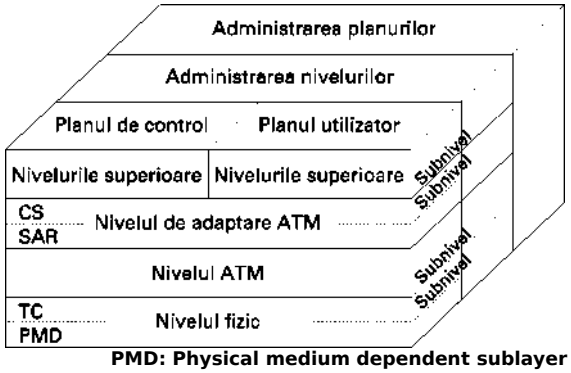
Reţelele ATM sunt organizate similar cu reţelele WAN tradiţionale, cu linii şi comutatoare (rutere). Cele mai des întâlnite viteze de lucru pentru reţelele ATM sunt 155 Mbps şi 622 Mbps, deşi sunt posibile şi viteze mai mari. Viteza de 155 Mbps a fost aleasă pentru că este foarte apropiată de viteza minimă obligatorie pentru transmisia de televiziune cu rezoluţie înaltă . Decizia de a alege viteza exactă de 155.52 Mbps a fost făcută pentru compatibilitatea cu sistemul de transmisie

SONET de la AT&T, care va fi studiat în cap. 2. Viteza de 622 Mbps a fost aleasă astfel încât să fie echivalentă cu transmisia simultană a 4 canale de 155 Mbps.

Modelul de referinţă ATM

ATM are propriul său model de referinţă, diferit de modelul OSI şi diferit de asemenea de modelul TCP/IP. Acest model este ilustrat în fig. l-32. El constă din trei niveluri – nivelul fizic, nivelul ATM şi nivelul de adaptare ATM – plus orice mai vrea utilizatorul să pună deasupra lor.

Nivelul fizic se ocupă de mediul fizic: voltaj, planificare la nivel de biţi şi diverse alte aspecte. ATM nu prescrie un set particular de reguli, dar spune în schimb că celulele ATM pot fi trimise direct prin cablu sau fibre optice sau pot fi, la fel de bine, împachetate în interiorul datelor din alte sisteme de transmisie. Cu alte cuvinte, ATM-ul a fost proiectat pentru a fi independent de mediul de transmisie.



CS: Convergence sublayer (subnivelul de convergenţă)
SAR: Segmentation and reassembly sublayer (subnivelul de segmentare şi reasamblare) TC: Transmission convergence sublayer (subnivelul de convergenţă a transmisiei)
PMD: Physical medium dependent sublayer (subnivelul dependent de mediul fizic)

Fig. l-32. Modelul de referinţă B-ISDN ATM.

Nivelul ATM se ocupă de celule şi de transportul celulelor. Nivelul defineşte structura unei celule şi spune ce reprezintă câmpurile celulelor. Tot el se ocupă şi de stabilirea şi eliberarea circuitelor virtuale. Controlul congestiei se realizează tot aici.

Deoarece cele mai multe aplicaţii nu vor să lucreze direct cu celule (deşi unele vor), deasupra nivelului ATM a fost definit un nivel care permite utilizatorilor să trimită pachete mai mari decât o celulă. Interfaţa ATM segmentează aceste pachete, transmite celulele individual şi le reasamblează la celălalt capăt. Acest nivel este **AAL** (**ATM Adaption Layer,** rom: nivelul de adaptare ATM).

Spre deosebire de cele două modele de referinţă anterioare, care erau bidimensionale, modelul ATM este definit ca fiind tridimensional, după cum se arată în fig. l-32. **Planul utilizator** se ocupă, printre altele, cu transportul datelor, controlul fluxului, corectarea erorilor. Prin contrast, sarcina **planului de control** este să trateze conexiunile. Funcţiile de administrare ale nivelurilor şi planurilor se referă la gestionarea resurselor şi coordonarea între niveluri.

Fiecare din nivelurile fizic şi AAL sunt împărţite în două subniveluri: un subnivel care face munca efectivă, la bază, şi un subnivel de convergenţă, deasupra, care

pune la dispoziţia nivelului situat peste el interfaţa adecvată. Funcţiile nivelurilor şi subnivelurilor sunt prezentate în fig. l-33.

Subnivelul **PMD** (**Physical Medium Dependent**, rom: dependent de mediul fizic) asigură interfaţa cu cablul propriu-zis. Acest subnivel transferă biţii şi se ocupă de planificarea transmisiei la nivel de biţi. În cazul unor companii telefonice şi a unor cabluri diferite, subnivelul va fi şi el diferit.

Nivel OSI	Nivel ATM	Subnivel ATM	
3/4	AAL	CS	Asigurarea interfeţei standard (convergenţei)
		SAR	Segmentarea şi reasamblarea
2/3	ATM		Controlul fluxului Generarea/extragerea antetelor din celule Administrarea circuitelor/căilor virtuale Multiplexarea/demultiplexarea celulelor
2	Fizic	TC	Decuplarea ratei celulelor Generarea şi verificarea sumelor de control din antete Generarea celulelor împachetarea/despachetarea celulelor din plic Generarea cadrelor
1		PMD	Temporizarea biţilor Accesul fizic la reţea

Fig. l-33. Nivelurile şi subnivelurile ATM şi funcţiile acestora.

Celălalt subnivel al nivelului fizic este subnivelul TC (Transmission Convergence, rom: convergenţa transmisiei). Când sunt transmise celulele, nivelul TC le expediază sub forma unui şir de biţi spre nivelul PMD. Acest lucru este uşor de făcut. La celălalt capăt, subnivelul TC primeşte de la subnivelul PMD un flux de biţi. Sarcina sa este să convertească acest flux de biţi într-un flux de celule pentru nivelul ATM. Subnivelul TC se ocupă de tot ce este necesar pentru a putea spune unde încep şi unde se termină celulele din fluxul de biţi. În modelul ATM această funcţionalitate este înglobată în nivelul fizic. În modelul OSI şi în majoritatea celorlalte reţele, încadrarea, adică transformarea unui flux oarecare de biţi într-o secvenţă de cadre sau de celule, este sarcina nivelului legătură de date. De aceea, în această carte vom discuta funcţia respectivă împreună cu nivelul legătură de date, nu cu nivelul fizic.

Aşa cum am menţionat mai devreme, nivelul ATM gestionează celulele, inclusiv generarea şi transportul lor. Mare parte din aspectele interesante ale ATM-ului apar aici. Nivelul ATM este un amestec între nivelurile legătură de date şi reţea de la OSI, dar nu este împărţit în subniveluri.

Nivelul AAL este împărţit într-un subnivel **SAR** (**Segmentation And Reassembly**, rom: segmentare şi reasamblare) şi un subnivel **CS** (**Convergence Sublayer**, rom: subnivel de convergenţă). Subnivelul inferior descompune pachetele în celule – la capătul la care are loc transmisia – şi le recompune la destinaţie. Subnivelul superior face posibile sistemele ATM care oferă diverse tipuri de servicii pentru diverse aplicaţii (de exemplu, transferul de fişiere şi sistemul video la cerere au cerinţe diferite privitoare la gestionarea erorilor, planificare etc.).

Deoarece se preconizează o evoluţie descendentă pentru reţelele ATM, ele nu vor fi discutate în continuare în această carte. Oricum, fiind instalate pe scară

destul de largă, vor fi în continuare folosite pentru câţiva ani buni. Pentru mai multe informaţii despre ATM, vedeţi (Dobrowski şi Grise, 2001; Gadeki şi Heckart, 1997).

1.5.3 Ethernet

Internet-ul şi ATM au fost proiectate pentru WAN. Oricum, multe companii, universităţi şi alte organizaţii au multe calculatoare care trebuie conectate. Această necesitate a dus la o dezvoltare rapidă a reţelelor locale. În această secţiune vom prezenta câteva lucruri despre cea mai populară dintre reţelele locale, şi anume Ethernet.

Povestea începe în primitivul Hawaii la începutul anilor 1970. În acest caz, „primitiv" poate fi interpretat ca „fără sistem de telefonie funcţional". Chiar dacă faptul că nu te deranjează telefonul cât e ziua de lungă poate să facă viaţa mai plăcută în vacanţă, această situaţie nu era foarte plăcută pentru cercetătorul Norman Abramson şi colegii săi de la Universitatea din Hawaii, care încercau să conecteze utilizatorii din mai multe insule aflate la distanţă la calculatorul principal din Honolulu. Şi cum varianta de a-şi trage singuri cablurile pe fundul Oceanului Pacific nu părea viabilă, a trebuit să se caute o altă soluţie.

Cea pe care au găsit-o a fost transmisia radio pe unde scurte. Fiecare terminal utilizator era echipat cu un mic sistem radio care avea două frecvenţe: Trimite (**upstream –** către calculatorul central) şi Primeşte (**downstream –** de la calculatorul central). Când utilizatorul dorea să contacteze calculatorul, trebuia doar să transmită un pachet care conţinea datele pe canalul Trimite. Dacă nu mai transmitea nimeni în acel moment, pachetul ajungea la calculatorul central şi i se dădea un răspuns pe canalul Primeşte. Dacă avea loc o dispută pentru canalul de transmisie, terminalul observa că nu primeşte confirmarea pozitivă pe canalul de recepţie şi trimitea din nou. Deoarece era un singur transmiţător pe canalul de primire (calculatorul central), aici erau imposibile coliziunile. Acest sistem, care a fost denumit ALOHANET, funcţiona destul de bine în condiţii de trafic redus, dar eşua de îndată ce traficul pe canalul de Transmisie era aglomerat.

Cam în acelaşi timp, un student pe nume Bob Metcalfe şi-a obţinut diploma de absolvire la M.I.T. şi s-a mutat pentru a obţine doctoratul la Harvard. În timpul studiilor sale, a ajuns sa cunoască lucrarea lui Abramson. A devenit atât de interesat în acest domeniu încât după ce a absolvit la Harvard, a decis să petreacă vara în Hawaii lucrând împreună cu Abramson, înainte de a începe lucrul la Xerox PARC (Palo Alto Research Center, rom: Centrul de Cercetare de la Palo Alto) . Când a ajuns la PARC, a descoperit că cercetătorii de acolo proiectaseră şi construiseră maşinile care mai târziu aveau să fie denumite calculatoare personale. Dar maşinile erau izolate. Folosind cunoştinţele pe care le acumulase în timpul lucrului petrecut cu Abramson, a proiectat şi implementat – împreună cu colegul său David Boggs – prima reţea locală de calculatoare (Metcalfe şi Boggs, 1976).

Au numit sistemul **Ethernet** după *luminiferous ether* (eter), prin care se credea odinioară că se propagă undele electromagnetice (În secolul 19, când fizicianul englez James Clerk Maxwell a descoperit că radiaţia electromagnetică poate fi

descrisă printr-o ecuaţie de undă, oamenii de ştiinţă au presupus că spaţiul trebuie să fie umplut cu un mediu eteric prin care aceste radiaţii se propagau. Numai după faimosul experiment Michelson-Morley din 1887 fizicienii au descoperit că radiaţia electromagnetică se poate propaga în vid).

Mediul de transmisie în acest caz era un cablu coaxial gros, având o lungime de până la 2.5 km (cu repetoare la fiecare 500m). Până la 256 de maşini pot fi ataşate sistemului prin transivere conectate direct în cablu. Un cablu cu mai multe maşini ataşate în paralel este numit **cablu multidrop** (multidrop cable). Sistemul funcţiona la 2.94 Mbps. O schiţă a arhitecturii sale este prezentată în fig. l-34. Ethernet-ul avea o îmbunătăţire majoră faţă de AOHANET: înainte să transmită, un calculator asculta mediul pentru a vedea dacă nu cumva este altcineva care transmite. Dacă exista deja o transmisie în curs, calculatorul se oprea şi aştepta încheierea transmisiei curente. Astfel, se evita interferenţa cu transmisiunile existente, ceea ce creştea semnificativ eficienţa sistemului. ALOHANET nu putea să funcţioneze în această manieră pentru că era imposibil pentru un terminal de pe o insulă să detecteze transmisia unui alt terminal de pe o altă insulă. Pe un cablu unic, această problem ă era rezolvată.

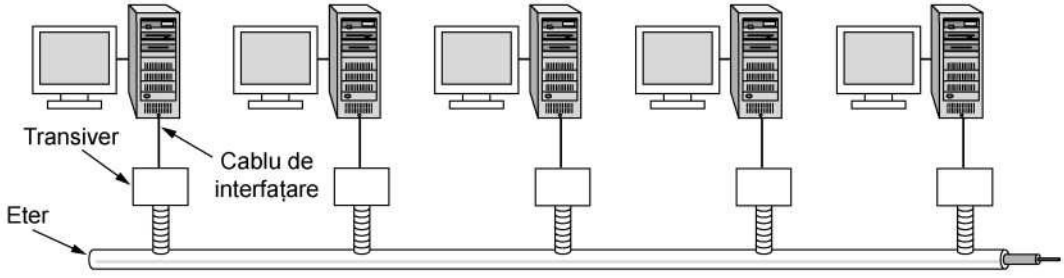


Fig. l-34. Arhitectura Ethenet-ului original

În ciuda faptului că fiecare calculator asculta mediul înainte să înceapă transmisia, exista în continuare o problemă: ce se întâmplă dacă două calculatoare aşteaptă amândouă încheierea transmisiei curente şi apoi pornesc propriile transmisii simultan? Soluţia este următoarea: fiecare calculator va asculta mediul şi în timpul propriei transmisii şi dacă detectează interferenţe, bruiază linia pentru a anunţa toţi transmiţătorii. Apoi se retrage şi aşteaptă un interval de timp generat aleator înainte să încerce din nou. Dacă apare o a doua coliziune, timpul de aşteptare se dublează, şi tot aşa, pentru a dispersa (în timp) transmisiile concurente oferind fiecăreia dintre ele şansa de a fi „servită” prima.

Ethernet-ul Xerox a avut un succes atât de mare încât DEC, Intel şi Xerox au colaborat pentru a schiţa un standard pentru o reţea Ethernet de 10 Mbps, denumit standardul DIX. Cu două modificări minore, acesta a devenit standardul IEEE 802.3 în anul 1983.

Din păcate pentru Xerox, compania avea deja reputaţia de a face invenţii (precum calculatorul personal) şi apoi să eşueze în valorificarea lor comercială, poveste spusă în *Fumbling the Future* (Smith şi Alexander, 1988). Şi pentru că Xerox nu a anunţat vreo intenţie de a face şi altceva cu Ethernet-ul – în afara

standardizării lui – Metcalfe şi-a format propria companie, 3Com, care urma să producă şi să vândă adaptoare Ethernet pentru PC. A vândut peste 100 de milioane.

Ethernet-ul a continuat să se dezvolte şi este încă în curs de dezvoltare. Noi versiuni, la 100 Mbps şi 1000 Mbps, ba chiar şi mai rapide au apărut deja. De asemenea, cablarea s-a îmbunătăţit, fiind adăugate şi alte facilităţi, precum comutarea (switching). Vom discuta în detaliu despre Ethernet în cap. 4.

În trecere, merită menţionat că Ethernet (IEEE 802.3) nu este singurul standard LAN. Comitetul a standardizat de asemenea Token Bus (Jeton pe Magistrală – 802.4) şi Token Ring (Jeton pe Inel – 802.5). Necesitatea de a avea trei standarde mai mult sau mai puţin incompatibile ţine mai mult de politică decât de tehnologie. La momentul standardizării, firma General Motors promova o reţea în care topologia era aceeaşi ca la Ethernet (un cablu liniar), dar calculatoarele obţineau dreptul la transmisie pe rând, prin transmiterea unui scurt pachet denumit **jeton (token)**. Un calculator putea să emită numai dacă era în posesia jetonului, fiind evitate astfel coliziunile. General Motors a

anunţat că această schemă era esenţială pentru fabricaţia de maşini şi nu era pregătită să se mişte de pe această poziţie. Dacă acest anunţ nu era susţinut, 802.4 nu ar fi existat.

Similar, IBM avea propriul favorit: reţeaua proprietară cu jeton în inel. De această dată, jetonul era transmis prin inel şi orice calculator care avea jetonul putea să transmită înainte de a repune jetonul în circulaţie în inel. Spre deosebire de 802.4, această schemă, standardizată ca 802.5, este încă folosită în birouri şi filiale ale IBM, dar practic nicăieri în afara IBM. Oricum, cercetarea avansează către o versiune gigabit, dar pare foarte puţin probabil ca această tehnologie să ajungă la nivelul Ethernet. Pe scurt, chiar dacă a fost cândva un război între Ethernet, Token Ring şi Token Bus, Ethernet a câştigat, în special pentru că a fost primul şi pentru că oponenţii săi nu era destul de buni.

1.5.4 Reţele fără fir: 802.11

Imediat după apariţia calculatoarelor portabile, mulţi utilizatori visau să intre cu calculatorul portabil personal într-un birou şi, miraculos, acesta să fie conectat la Internet. În consecinţă, mai multe grupuri de studiu am început să caute soluţii pentru a atinge acest scop. Cea mai practică abordare era echiparea biroului şi a calculatorului cu transmiţătoare şi emiţătoare radio cu rază mică de acţiune pentru a le permite să comunice. Această variantă a dus rapid la comercializarea soluţiilor de reţele locale fără fir de către diverse companii.

Problema era că dintre aceste variante nu se găseau două compatibile. Această proliferare a standardelor însemna că un calculator care era echipat cu un radio marca X nu putea să se conecteze în reţeaua unui birou dacă acesta era echipat cu o staţie de la firma Y. În cele din urmă, comunitatea industrială a decis că ar trebui impus un standard pentru LAN fără fir. Astfel, comitetul IEEE care standardizase şi LAN-urile cu cablu a primit ca sarcină să schiţeze un standard pentru reţele LAN fără fir. Standardul astfel creat s-a numit 802.11. O denumire mai bine cunoscută în argou este **WiFi**. Este un standard important şi merită tot respectul, astfel că ne vom referi la el cu numele oficial, 802.11.

Standardul propus trebuia să lucreze în două moduri:

1. În prezenţa unei staţii de bază
2. În absenţa unei staţii de bază

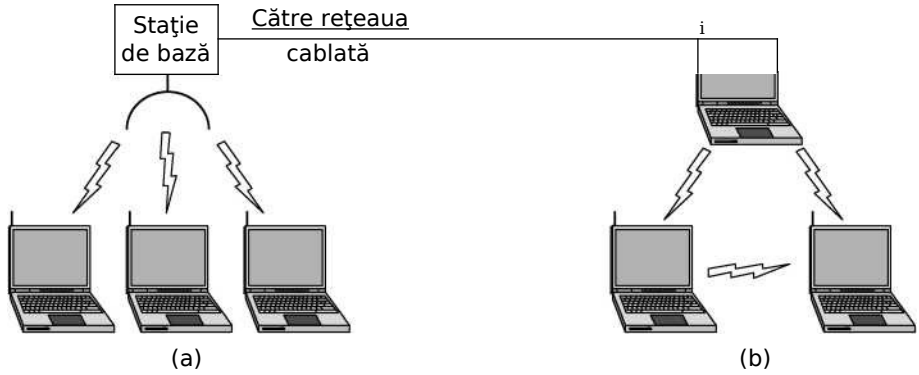


Fig. l-35. (a) Reţele fără fir cu staţie de bază. (b) Conectare

În primul caz, toate comunicaţiile urmau să aibă loc prin intermediul staţiei de bază, denumită **punct de acces** (**access point**) 802.11. În cel de-al doilea caz, calculatoarele urmau să comunice direct unul cu celălalt. Acest mod este uneori denumit **conectare ad-hoc** (**ad-hoc networking**). Un exemplu tipic este cel al utilizatorilor care se află într-o cameră care nu este echipată cu o staţie de bază, calculatoarele lor comunicând direct. Aceste două moduri sunt ilustrate în fig. l-35.

Prima decizie a fost cea mai simplă : cum să se numească . Toate celelalte standarde LAN aveau numere cum sunt 802.1, 802.2, 802.3, până la 802.10. Aşa că noul standard de LAN fără fir s-a numit 802.11. Restul a fost mai dificil de realizat.

În particular, câteva dintre obiectivele care trebuiau atinse erau : găsirea unei benzi de frecvenţe care să fie disponibilă, de preferinţă la nivel mondial; tratarea faptului că semnalele radio au o rază de acţiune limitată; asigurarea menţinerii confidenţialităţii utilizatorului; tratarea problemei duratei limitate de lucru a bateriei; considerarea eventualelor efecte pe care sistemul le putea avea asupra oamenilor (provoacă undele radio cancer?); înţelegerea implicaţiilor portabilităţii calculatoarelor; şi, în final, construirea unui sistem cu lărgime de bandă suficientă pentru a fi viabil din punct de vedere economic.

La momentul în care s-a început procesul de standardizare (la mijlocul anilor 1990), Ethernet-ul domina deja domeniul reţelelor locale, aşa încât comitetul a decis să facă noul standard 802.11 compatibil Ethernet începând de deasupra nivelului legătură de date. Mai exact, ar trebui să se poate transmite un pachet IP într-un LAN fără fir în aceeaşi manieră în care un pachet IP este transmis prin Ethernet. Desigur, la nivelurile Fizic şi Legătură de date apar anumite diferenţe inerente faţă de Ethernet şi ele trebuie considerate de către standard.

Mai întâi, un calculator din Ethernet va asculta eterul înainte de a transmite. Numai dacă acesta este liber calculatorul va începe transmisia. În cazul reţelelor LAN fără fir, această idee nu funcţionează prea bine. Pentru a vedea de ce, analizaţi fig. l-36. Să presupunem că A transmite către B, dar raza de acţiune a lui A este prea mică pentru a îl acoperi şi pe C. Atunci când C vrea să transmită, el poate asculta mediul înainte să înceapă, dar faptul că nu aude nimic nu înseamnă că transmisia lui va reuşi. Standardul 802.11 trebuia să rezolve şi această problemă.

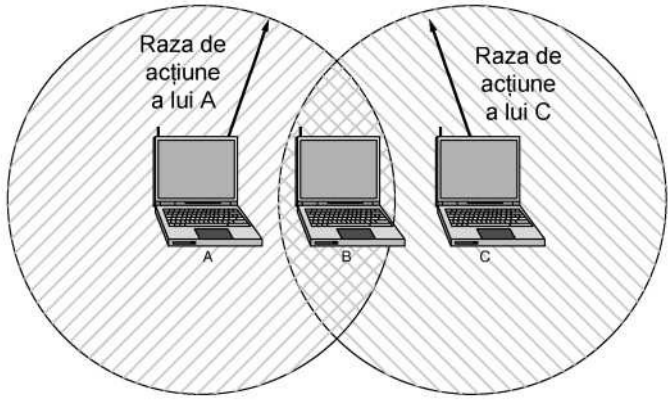


Fig.1-36. Raza de acţiune a unui singur radio poate să nu acopere întregul sistem.

O a doua problemă care trebuia rezolvată era aceea că semnalul radio poate fi reflectat de anumite obiecte solide şi deci poate fi recepţionat de mai multe ori (pe diverse căi). Această interferenţă duce la ceea ce se numeşte **disipare pe mai multe căi** (**multipath fading**).

Cea de-a treia problemă era că o mare parte din aplicaţii nu erau conştiente de mobilitatea calculatoarelor. De exemplu, multe dintre editoarele de texte aveau o listă de imprimante dintre care una putea fi aleasă pentru tipărirea documentului. Atunci când calculatorul rulează în afara mediului său obişnuit, într-un mediu nou, lista de imprimante implicite nu mai este validă.

Cea de-a patra problemă se referea la mutarea calculatorului portabil din raza de acţiune a unei staţii de bază în raza altei staţii de bază. Într-un fel sau altul, trebuie găsit ă o soluţie de preda- re/primire între cele două staţii de bază. Deşi aceast ă problemă apare şi la nivelul telefoanelor mobile, ea nu apare la Ethernet şi nu avea o soluţie la momentul respectiv. Mai exact, reţeaua const ă din mai multe celule, fiecare cu propria staţie de bază, conectate prin Ethernet, după cum se poate vedea în fig. l-37. Din exterior, sistemul trebuie să arate ca o singură reţea Ethernet. Conexiunea dintre sistemele 802.11 şi lumea exterioară se numeşte **portal** (**portal**).

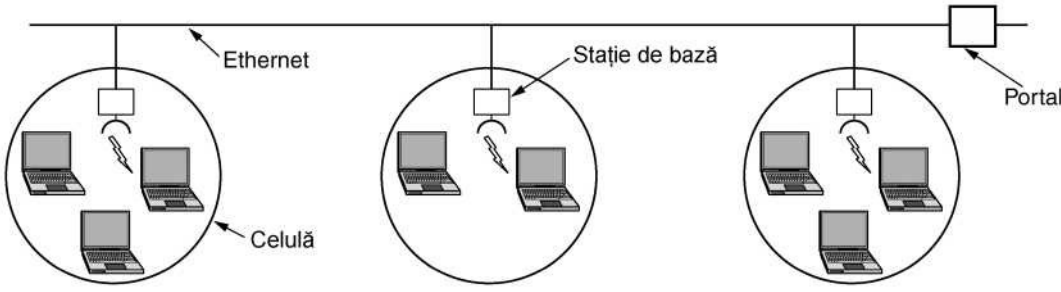


Fig.1-37. O reţea 802.11 cu mai multe celule

După o oarecare muncă, comitetul a obţinut o variantă de standard în 1997, variantă care adresa aceste probleme şi altele asemănătoare. Reţelele locale fără fir

pe care standardul le propunea puteau funcţiona la 1 Mbps sau 2 Mbps. Aproape imediat, utilizatorii au început să se plângă de viteza prea scăzută şi s-a pornit o nouă campanie pentru obţinerea unor standarde mai rapide. În cadrul comitetului a avut loc o ruptură, ceea ce a dus la apariţia a două standarde în 1999. Standardul 802.11a foloseşte o bandă de frecvenţă mai largă şi poate ajunge la viteze mai mari de 54 Mbps. Standardul 802.11b foloseşte aceeaşi banda ca şi 802.11, dar foloseşte o tehnică de modulare diferită şi poate ajunge la 11 Mbps. Unii văd în aceasta un amănunt important la nivel psihologic, pentru că 11 Mbps este o viteză mai mare decât a Ethernet-ului original, cu cablu. Este foarte probabil ca standardul original 802.11 de 1 Mbps să moară în curând, dar nu se ştie care dintre noile standarde va ieşi învingător.

Pentru a face lucrurile încă mai complicate decât erau, comitetul 802 a venit cu o nouă variantă, 802.11g, care foloseşte tehnica de modulare folosită şi de 802.11a, dar banda de frecvenţă a lui 802.11b. Vom reveni în detaliu la 802.11 în cap. 4.

Faptul ca 802.11 urmează să ducă la o revoluţie în lumea calculatoarelor şi a accesului la Internet este mai presus de orice îndoială. Aeroporturile, gările, hotelurile, magazinele mari şi universităţile îl implementează foarte curând. Chiar şi cafenelele aflate într-o perioadă de creştere a afacerilor instalează reţele 802.11 pentru ca grupurile de tineri rebeli să poată naviga pe Web în timp ce îşi savurează cafelele cu lapte. Este foarte probabil ca 802.11 să aibă asupra Internet-ului acelaşi efect pe care lau avut portabilele în lumea calculatoarelor: să-l facă mobil.

1.6 STANDARDIZAREA REŢELELOR

În prezent există numeroşi producători şi furnizori, fiecare cu propriile idei despre cum ar trebui realizate reţelele. În lipsa coordonării, ar fi un haos complet şi utilizatorii nu ar putea face nimic. Singura soluţie este să se convină asupra unor standarde de reţea.

Standardele nu numai că permit diverselor calculatoare să comunice între ele, ci sporesc totodată piaţa pentru produsele care aderă la un anumit standard, cu următoarele consecinţe: producţie de masă, profituri financiare, implementări VLSI şi alte beneficii care duc la scăderea preţurilor şi la acceptarea şi mai largă a respectivelor produse. În secţiunile următoare vom arunca o privire asupra importantei, dar puţin cunoscutei, lumi a standardizării internaţionale.

Standardele fac parte din două categorii: de facto şi de jure. Standardele **de facto** (expresia latină pentru ,,de fapt”) sunt acelea care pur şi simplu au luat fiinţă, fără să existe vreun plan oficial. Deoarece zeci de producători au decis să copieze aproape identic maşinile IBM, PC-ul IBM şi succesorii săi reprezintă standarde de facto pentru calculatoarele birourilor mici şi pentru cele casnice. În secţiile de informatică ale facultăţilor, UNIX este standardul de facto pentru sisteme de operare.

Standardele **de jure** (expresia latină pentru ,,de drept”) sunt, prin contrast, standarde legale, adoptate de un anumit organism de standardizare autorizat. Autorităţile de standardizare internaţionale sunt, în general, împărţite în două clase: organizaţii stabilite prin tratate între guvernele naţionale şi organizaţii voluntare neguvernamentale. În domeniul standardelor pentru reţele de

calculatoare există câteva organizaţii din fiecare categorie. În continuare vom discuta despre aceste organizaţii.

1.6.1 Who's Who în lumea telecomunicaţiilor

Statutul legal al companiilor telefonice de pe glob variază considerabil de la ţară la ţară. La una din extreme se situează Statele Unite, care au 1500 de firme de telefonie private. Înainte să fie divizată, în 1984, AT&T, cea mai mare corporaţie din lume la vremea aceea, domina scena complet. AT&T furniza servicii telefonice pentru aproximativ 80 la sută din telefoanele Americii, răspândite pe jumătate din întinderea sa, în timp ce toate celelalte firme asigurau servicii pentru restul clienţilor (rurali, în majoritatea lor). De la divizarea sa, AT&T continuă să furnizeze servicii de lungă distanţă, dar acum o face în concurenţă cu alte firme. Cele şapte Companii Regionale Bell în care a fost împărţit AT&T-ul, precum şi alte numeroase firme independente, oferă servicii de telefonie locală şi celulară. Datorită fuziunilor frecvente şi a altor modificări de acest tip, această industrie este într-o continuă mişcare.

Firmele americane furnizoare de servicii de comunicaţii pentru public sunt numite **companii telefonice publice**. Ofertele şi preţurile lor sunt descrise printr-un document numit **tarif.** Acesta trebuie să fie aprobat de Comisia Federală de Comunicaţii, care se ocupă de traficul dintre statele SUA şi de traficul internaţional, precum şi de către comisiile publice de stat pentru traficul în interiorul său.

La cealaltă extremă se află ţările în care guvernul are un monopol complet asupra tuturor mijloacelor de comunicaţie: poşta, telegraful, telefonul şi, de multe ori, chiar radioul şi televiziunea. Cea mai mare parte a lumii se încadrează în această categorie. În unele cazuri, autoritatea de telecomunicaţii este o companie naţionalizată, în altele, este o simplă filială a guvernului, cunoscută de obicei sub numele de **PTT** (**Post, Telegraf & Telephone** administration). Tendinţa actuală în lumea întreagă este către liberalizare şi competiţie şi împotriva monopolului guvernamental. Majoritatea ţărilor europene şi-au privatizat – mai mult sau mai puţin – sistemele PTT, dar peste tot acest proces este lent.

Din cauza tuturor acestor diverşi furnizori de servicii este nevoie de o compatibilitate la scară mondială. Compatibilitatea asigură faptul că oamenii (şi calculatoarele) dintr-o ţară pot să-şi apeleze partenerii din altă ţară. La drept vorbind, această necesitate există de mult timp. În 1865, reprezentanţi ai multor guverne din Europa s-au întâlnit pentru a forma predecesorul actualului **ITU** (**International Télécommunication Union**, rom: Uniunea Internaţională de Telecomunicaţii). Sarcina Uniunii era standardizarea telecomunicaţiilor internaţionale, care la vremea aceea însemnau telegrafia. Chiar de atunci, era clar că dacă jumătate din ţări foloseau codul Morse şi cealaltă jumătate foloseau un cod diferit, atunci vor apare probleme. Când au apărut serviciile de telefonie internaţională, ITU a preluat de asemenea şi sarcina standardizării telefoniei (telephony – pronunţat şi te-LEF-ony). În 1947 ITU a devenit o agenţie a Naţiunilor Unite. ITU are trei sectoare principale:

1. Sectorul de Radiocomunicaţii (ITU-R).
2. Sectorul de Standardizare a Telecomunicaţiilor (ITU-T).
3. Sectorul de dezvoltare (ITU-D).

ITU-R se ocupă de alocarea frecvenţelor internaţionale de radio către grupurile concurente interesate. Ne vom referi mai întâi la ITU-T, care se ocupă de sistemele de telefonie şi de comunicare de date. Din 1956 până în 1993, ITU-T a fost cunoscut ca **CCITT**, un acronim pentru numele său francez: Comité Consultatif International Télégraphique et Téléfonique. La 1 martie 1993, CCITT a fost reorganizat în scopul de a deveni mai puţin birocratic şi a fost redenumit pentru a reflecta noul său rol. Atât ITU-T cât şi CCITT au dat recomandări în domeniul telefoniei şi comunicaţiilor de date. Deşi, începând cu 1993, recomandările poartă eticheta ITU-T, recomandările CCITT, de genul CCITT X.25, mai sunt încă frecvent întâlnite.

ITU-T are patru clase de membri:

1. Guverne naţionale
2. Membri sectoriali
3. Membri asociaţi
4. Agenţii de reglementare

ITU-T are aproximativ 200 de membri guvernamentali, incluzând aproape fiecare membru al Naţiunilor Unite. Pentru că SUA nu are un sistem PTT, altcineva trebuia să o reprezinte în cadrul ITU-T. Această sarcină a revenit Departamentului de Stat, probabil pe principiul că ITU-T are de-a face cu ţări străine, tocmai specialitatea acestui departament. Sunt aproximativ 500 de membri sectoriali, incluzând aici companiile de telefonie (AT&T, Vodafone, WorldCom), producătorii de echipamente de telecomunicaţii (Cisco, Nokia, Nortel), producătorii de echipamente de calcul (Compaq, Sun, Toshiba), producătorii de cipuri (Intel, Motorola, TI), companii media (AOL Time, Warner, CBS, Sony) şi alte companii direct interesate (Boeing, Samsung, Xerox). Diverse organizaţii ştiinţifice non-profit, precum şi consorţii industriale sunt de asemenea membri sectoriali (IFIP, IATA). Membrii asociaţi sunt organizaţii mai mici care sunt interesate într-un anumit grup de studiu. Agenţiile de reglementare sunt reprezentate de oamenii care supraveghează lumea afacerilor în telecomunicaţii, cum este de exemplu US Federal Communications Comission (Comisia Federală pentru Comunicaţii).

Sarcina pe care o are ITU-T este de a face recomandări tehnice asupra interfeţelor din telefonie, telegrafie şi comunicaţii de date. Acestea devin deseori standarde recunoscute internaţional; de exemplu, V.24 (cunoscut în Statele Unite şi ca EIA RS-232), specifică amplasarea şi semnificaţia pinilor din conectorul folosit de majoritatea terminalelor asincrone şi de modemurile externe.

Nu trebuie uitat că recomandările date de ITU-T sunt numai sugestii tehnice, pe care guvernele le pot adopta sau ignora, după cum doresc (pentru că guvernele sunt asemenea băieţilor de 13 ani – nu reacţionează prea bine dacă li se dau ordine). În practică, o ţară care doreşte să adopte un standard de telefonie diferit de cel utilizat în restul lumii este liberă să o facă, dar o face cu preţul izolării de toate celelalte ţări. Lucrul acesta poate să meargă în cazul Coreei de Nord, dar în

altă parte ar fi o adevărată problemă. Fantezia de a numi standardele ITU-T „recomandări” a fost şi este necesară pentru a calma forţele naţionaliste din multe ţări.

Adevărata muncă de la ITU-T se desfăşoară în grupuri de studiu, care uneori cuprind chiar şi 400 de persoane. Momentan sunt 14 grupuri de studiu, care acoperă subiecte de la facturarea serviciilor telefonice până la serviciile multimedia. Pentru ca până la urmă munca să aibă un rezultat, Grupurile de Studiu se împart în Echipe de Lucru, care se împart la rândul lor în Echipe de Experţi, care, la rândul lor, se împart în grupuri ad-hoc. Birocraţie a fost, birocraţie rămâne.

În pofida tuturor acestor lucruri, ITU-T reuşeşte să ducă la bun sfârşit ceea ce are de făcut. De la fondarea sa, a realizat mai bine de 3000 de recomandări, care ocupă peste 60.000 de pagini. Multe dintre acestea sunt folosite pe scară largă în practică. De exemplu, standardul V.90 56-Kbps pentru modemuri este o recomandare a ITU.

Pe măsură ce telecomunicaţiile desăvârşesc tranziţia – începută în anii 1980 – de la un caracter strict naţional la un caracter complet global, standardele vor deveni din ce în ce mai importante şi tot mai multe organizaţii vor dori să devină implicate în producerea acestora. Pentru mai multe informaţii privind ITU, a se vedea (Irmer, 1994).

1.6.2 Who’s Who în lumea standardelor internaţionale

Standardele internaţionale sunt produse de **ISO** (**Internaţional Standards Organization**[****], rom: Organizaţia Internaţională de Standardizare), o organizaţie voluntară, neguvernamentală fondată în 1946. Membrii săi sunt organizaţiile naţionale de standardizare din cele 89 de ţări membre. Aceşti membri cuprind ANSI (S.U.A.), BSI (Marea Britanie), AFNOR (Franţa), DIN (Germania) şi încă 85 de alte organizaţii.

ISO produce standarde referitoare la un număr vast de subiecte, începând cu piuliţe şi şuruburi şi terminând cu vopsirea stâlpilor de telefon [pentru a nu menţiona aici boabele de cacao (ISO 2451), plasele de pescuit (ISO 1530), lenjeria de damă (ISO 4416) şi alte câteva subiecte la care nu v-aţi putea gândi ca subiecte de standarde]. În total au fost create peste 5000 de standarde, inclusiv standardele OSI. ISO are aproape 200 de Comitete Tehnice (Technical Committees – TC), numerotate în ordinea creării lor, fiecare comitet ocupându-se de un subiect specific. TC1 se ocupă de piuliţe şi şuruburi (standardizarea înclinării filetelor). TC97 se ocupă de calculatoare şi prelucrarea informaţiei. Fiecare TC are subcomitete (SC-uri) împărţite în grupe de lucru (Work Groups – WG).

Munca propriu-zisă se desfăşoară în principal în WG-uri, prin intermediul a peste 100.000 de voluntari din întreaga lume. Mulţi dintre aceşti ,,voluntari” sunt puşi să lucreze la probleme ale ISO de către patronii lor, ale căror produse sunt standardizate. Alţii sunt oficiali guvernamentali dornici să vadă că modalitatea de a face lucrurile în ţara lor devine standardul internaţional. În multe WG-uri sunt

****Adevaratul nume pentru ISO este International Organization for Standardization (n.a.)

activi, de asemenea, experţi academici. În ceea ce priveşte standardele din telecomunicaţii, ISO şi ITU-T cooperează frecvent, (ISO este un membru al ITU-T) în ideea de a evita ironia a două standarde internaţionale oficiale şi mutual incompatibile.

Număr	Subiect
802.1	Principiile generale şi arhitectura LAN-urilor
802.2	Controlul legăturii logice
802.3 *	Ethernet
802.4	TokenBus (Jeton pe Magistrală – utilizat câtva timp în fabrici)
802.5	TokenRing (Jeton în Inel – contribuţia IBM la lumea LAN)
802.6 sL	Coadă dual ă, magistral ă dual ă (reţea metropolitană timpurie)
802.7	Grupul de consiliere tehnică pe probleme de tehnologii de bandă largă
802.8 Ţ	Grupul de consiliere tehnică pe probleme de tehnologii de fibră optică
802.9	LAN-uri izocrone pentru aplicaţii de timp real
802.10	LAN-uri virtuale şi securitate
802.11 *	LAN-uri fără fir
802.12	Prioritatea cererilor (AnyLAN de la HP)
802.13	Număr cu ghinion. Nimeni nu l-a vrut
802.14	Modemuri de cablu (decedat: un consorţiu industrial a abordat înainte domeniul)
802.15*	Reţele personale (Bluetooth)
802.16*	Comunicaţii fără fir în bandă largă
802.17	Inel activ de pachete

Fig. l-38. Grupurile de lucru ale 802.
Cele importante sunt marcate cu *. Cele marcate cu hibernează.
Cele marcate cu ţ au renunţat şi s-au desfiinţat.

Reprezentantul S.U.A. în ISO este **ANSI** (**American National Standards Institute**, rom: Institutul Naţional American de Standarde), care, în pofida numelui său, este o organizaţie privată neguvernamentală şi nonprofit. Membrii săi sunt producători, companii telefonice publice şi alte părţi interesate. Standardele ANSI sunt frecvent adoptate de ISO ca standarde internaţionale.

Procedura utilizată de ISO pentru adoptarea standardelor este concepută astfel încât să se obţină un consens cât mai larg posibil. Procesul începe când una din organizaţiile naţionale de standardizare simte nevoia unui standard internaţional într-un anumit domeniu. În acel moment, se formează un grup de lucru care vine cu un **CD** (**Committee Draft**, rom: proiect de comitet). CD-ul circulă apoi pe la toate organizaţiile membre, care au la dispoziţie 6 luni pentru a-l supune criticilor. Dacă se primeşte aprobarea din partea unei majorităţi substanţiale, atunci se produce un document revizuit, numit **DIS** (**Draft International Standard**, rom: proiect de standard internaţional), care va circula în scopul de a fi comentat şi votat. Pe baza rezultatelor din această rundă, se pregăteşte, se aprob ă şi se publică textul final al respectivului **IS** (**International Standard**, rom: standard internaţional). În domeniile foarte controversate, un CD sau un DIS pot să treacă prin câteva versiuni

înainte de a obţine suficiente voturi şi întregul proces poate dura ani de zile.

NIST (**Naţional Institute of Standards and Technology**, rom: Institutul Naţional de Standarde şi Tehnologie) este o agenţie a Departamentului pentru Comerţ al Statelor Unite. NIST a fost cunoscut anterior sub numele de Biroul Naţional de Standarde. El produce standarde care sunt obligatorii pentru achiziţiile făcute de guvernul U.S.A., mai puţin pentru cele care privesc Departamentul de Apărare, acesta având propriile sale standarde.

Un alt actor important din lumea standardelor este **IEEE** (**Institute of Electrical and Electronics Engineers**, rom: Institutul Inginerilor Electricieni şi Electronişti), cea mai mare organizaţie profesională din lume. Suplimentar faţă de producerea a zeci de jurnale şi organizarea a numeroase conferinţe în fiecare an, IEEE are un grup de standardizare care dezvoltă standarde în domeniul ingineriei electrice şi tehnicii de calcul. Comitetul IEEE 802 a standardizat mai multe tipuri de reţele locale. Vom studia o parte dintre rezultatele sale ceva mai târziu în această carte. Munca efectivă este făcută de o sumă de grupuri de lucru, care sunt prezentate în fig. l-38. Rata de succes a diverselor grupuri ale 802 a fost scăzută, aşadar chiar dacă ai un număr de forma 802.x, aceasta nu este o garanţie a succesului. Dar impactul poveştilor de succes (în special 802.3 şi 802.11) a fost enorm.

1.6.3 Who’s Who în lumea standardelor Internet

Internet-ul mondial are propriile sale mecanisme de standardizare, foarte diferite de cele ale ITU-T şi ISO. Diferenţa poate fi rezumată grosier spunând că lumea care vine la întâlnirile pentru standardizare ale ITU şi ISO poartă costum. Lumea care vine la întâlnirile pentru standardizarea Internet-ului poartă blugi (iar dacă se întâlnesc la San Diego poartă pantaloni scurţi şi tricouri).

La întâlnirile organizate de ITU-T şi ISO e plin de oficiali ai unor corporaţii şi de funcţionari guvernamentali pentru care standardizarea reprezintă meseria lor. Ei privesc standardizarea ca un lucru bun şi îşi dedică vieţile acestui scop. Lumea implicată în Internet, pe de altă parte, preferă, ca principiu de bază, anarhia. Oricum, dacă sute de milioane de oameni îşi văd fiecare numai de treaba lor, este puţin probabil să apară vreo modalitate de comunicare. De aceea, standardele, deşi regretabile, apar ocazional ca fiind necesare.

Când a fost creat ARPANET-ul, DoD-ul a înfiinţat un comitet neoficial care să îl supravegheze. În 1983 comitetul a fost redenumit **IAB** (**Internet Activities Board**, rom: Consiliul Activităţilor Internet) şi a primit o misiune ceva mai amplă: să fie atent ca cercetătorii implicaţi în ARPANET şi Internet să se mişte, mai mult sau mai puţin, în aceeaşi direcţie – o activitate care ar putea fi asemănată cu „păstoritul” pisicilor. Semnificaţia acronimului ,,IAB” a fost schimbată mai târziu în **Internet Architecture Board** (Consiliul Arhitecturii Internet).

Fiecare din cei aproximativ 10 membri ai IAB-ului conducea un departament care se ocupa de o anumită problemă importantă. IAB-ul se întâlnea de câteva ori pe an pentru a discuta rezultatele şi a trimite informări către DoD şi NSF, care asigurau la acea vreme majoritatea fondurilor. Când era nevoie de un nou standard (de exemplu, un nou algoritm de dirijare), membrii IAB îl luau în discuţie şi apoi

anunţau schimbarea, astfel ca absolvenţii facultăţilor – care erau sufletul muncii de programare – să îl poată implementa. Comunicările erau puse la dispoziţie printr-o serie de rapoarte tehnice, numite **RFC**-uri (**Request For Comments**, rom: cereri pentru comentarii). RFC-urile sunt memorate on-line şi pot fi citite de oricine este interesat de ele la adresa *www.ietf.org/rfc*. RFC-urile sunt numerotate în ordinea cronologică a creării lor. Până acum există peste 3000. Ne vom referi la multe dintre ele în cursul acestei cărţi.

În 1989 Internet-ul crescuse atât de mult, încât acest stil informal nu mai putea funcţiona. Multe firme vindeau la acea vreme produse TCP/IP şi nu erau dispuse să le modifice doar pentru că zece cer

cetători se gândiseră la o idee mai bună. În vara anului 1989, IAB a fost reorganizat. Cercetătorii au fost transferaţi la **IRTF** (**Internet Research Task Force**, rom: Departamentul de Cercetare Internet), care a fost pus în subordinea IAB-ului, alături de **IETF** (**Internet Engineering Task Force**, rom: Departamentul de Inginerie Internet). IAB-ul a fost repopulat cu persoane care reprezentau un palier de organizaţii mai larg decât stricta comunitate a cercetătorilor. La început a fost un grup care se auto- perpetua: membrii erau activi pe o perioadă de 2 ani, iar membrii noi erau selectaţi de către membrii mai vechi. Mai târziu, a fost înfiinţată **Societatea Internet** (**Internet Society**), care reunea oameni interesaţi de Internet. Societatea Internet este, prin urmare, comparabilă într-un sens cu ACM sau IEEE. Societatea este administrată de un comitet ales, iar comitetul desemnează membrii IAB.

Ideea acestei divizări a fost ca IRTF să se concentreze asupra cercetării pe termen lung, iar IETF să se ocupe de probleme inginereşti pe termen scurt. IETF a fost împărţit în grupuri de lucru, fiecare cu o problemă specifică de rezolvat. Iniţial, preşedinţii grupurilor de lucru s-au reunit într- un comitet de organizare, în scopul de a coordona munca inginerească ce le revenea. Preocupările grupurilor de lucru includeau aplicaţii noi, informaţii de la utilizatori, integrare OSI, dirijare şi adresare, securitate, administrare de reţea, standarde. În final s-au format atât de multe grupuri de lucru (mai mult de 70), încât ele au fost grupate pe domenii, iar comitetul de organizare s-a constituit din preşedinţii domeniilor.

În plus, a fost adoptat un proces de standardizare mai formal, preluat după modelul ISO. Pentru a deveni un standard propus (**Proposed Standard**), ideea fundamentală trebuie să fie complet explicată într-un RFC şi să prezinte destul interes din partea comunităţii pentru a merita să fie luată în considerare. Pentru a avansa la stadiul de proiect de standard (**Draft Standard**), este necesară o implementare de lucru care să fi fost testată în amănunţime de către două situri independente, timp de cel puţin 4 luni. Dacă IAB-ul este convins că ideea e bună şi că programul funcţionează, atunci poate să declare RFC-ul respectiv ca fiind un Standard Internet. Unele Standarde Internet au devenit standarde ale DoD-ului (MIL-STD), fiind, prin urmare, obligatorii pentru furnizorii DoD-ului. David Clark a făcut odată o remarcă devenită celebră privitoare la standardizarea Internet-ului, care ar consta din „consens aproximativ şi programe care merg.”

1.7 UNITĂŢI DE MĂSURĂ

Pentru a ne feri de orice confuzie, merită să precizăm de la bun început că în această carte, ca şi în lumea ştiinţei calculatoarelor în general, vor fi folosite unităţile metrice în locul unităţilor tradiţionale englezeşti (sistemul furlong-stone-fortnight4). Principalele prefixe metrice sunt precizate în fig. l-39. Ale sunt în general abreviate folosindu-se prima literă, cu unităţile mai mari ca 1 scrise cu majuscule (KB, MB etc.). O excepţie (din motive istorice) este Kbps (kilobits per second) pentru kilobiţi pe secundă. Astfel, o linie de comunicaţie de 1 Mbps

transmite $10^{††††}$ ‡‡‡ §§§ biţi/secundă, în timp ce pentru 100 ps (psec), ceasul bate la fiecare 10^{-10} secunde. Deoarece denumirile mili şi micro încep amândouă cu litera „m”, trebuia făcută o alegere. În mod normal, „m” este folosit pentru mili, iar „p” (litera greacă *miu)* este folosit pentru micro.

Exp.	Explicit	Prefix	Exp.	Explicit	Prefix
10^{3}	0.001	milli	10^{3}	1,000	Kilo
10^{6}	0.000001	micro	10^{6}	1,000,000	Mega
10^{-9}	0.000000001	nano	10^{9}	1,000,000,000	Giga
10^{-12}	0.000000000001	pico	10^{12}	1,000,000,000,000	Tera
10^{-15}	0.000000000000001	femto	10^{15}	1,000,000,000,000,000	Peta
10^{-18}	0.000000000000000001	atto	10^{18}	1,000,000,000,000,000,000	Exa
10^{-21}	0.000000000000000000001	zepto	10^{21}	1,000,000,000,000,000,000,000	Zetta
10^{-24}	0.000000000000000000000001	yocto	10^{24}	1,000,000,000,000,000,000,000, 000	Yotta

Fig. l-39. Principalele prefixe metrice

Este de asemenea important să subliniem că pentru măsurarea dimensiunilor memoriei, discurilor, fişierelor şi a bazelor de date se obişnuieşte folosirea acestor unităţi, deşi ele au valori uşor modificate. Astfel, kilo reprezintă 2^{10} (1024) şi nu de 10^{3} (1000), pentru că volumului memoriilor sunt întotdeauna puteri ale lui doi. Deci, o memorie de 1 KB are 1024 de octeţi, nu 1000. Similar, o memorie de 1 MB are 2^{20} (1.048.576) octeţi, o memorie de 1 GB are 2^{30} octeţi (1.073.741.824), iar o bază de date de 1 TB are 2^{40} (1.099.511.627.776) octeţi. Oricum, o linie de comunicaţie de 1 Kbps transmite 1000 de biţi pe secundă şi o reţea locală de 10 Mbps rulează la 10.000.000 biţi/secundă, deoarece aceste unităţi nu sunt puteri ale lui 2. Din păcate, mulţi oameni tind să amestece aceste două sisteme, în special pentru capacitatea discurilor. Pentru a evita orice ambiguitate, în această carte vom folosi simbolurile KB, MB, GB pentru 2^{10}, 2^{20}, 2^{30}, şi simbolurile Kbps, Mbps şi Gbps pentru 10^{3}, 10^{6} şi 10^{9} biţi pe secundă, respectiv.

1.8 RESTUL CĂRŢII ÎN REZUMAT

Cartea de faţă discută atât principiile cât şi practica interconectării calculatoarelor. Majoritatea capitolelor încep printr-o discuţie a principiilor relevante, urmată de un număr de exemple care ilustrează principiile respective. Aceste exemple sunt în general preluate din Internet şi din reţele fără fir deoarece acestea sunt importante şi diferite. Acolo unde este relevant, vor fi date şi alte exemple.

---

†††† furlong = jumătate de milă

‡‡‡‡stone = 6,350kg

§§§§fortnight = 2 săptămâni

Cartea este structurată în concordanţă cu modelul hibrid din fig. l-24. Începând cu cap. 2, pornim la drum de la bază în sus, de-a lungul ierarhiei de protocoale. Cap. 2 prezintă cadrul pentru studierea domeniului comunicaţiilor de date. Capitolul acoperă diferite subiecte: transmisii analogice şi digitale, multiplexare, comutare, sistemul telefonic trecut, actual şi viitor. Acoperă sisteme de transmisie cu cablu, fără cablu şi prin satelit. Acest material se referă la nivelul fizic, dar noi ne vom ocupa numai de aspectele arhitecturale, nu de cele privitoare la echipamente. Sunt discutate, de asemenea, câteva exemple de niveluri fizice, cum ar fi reţeaua cu comutare a telefoniei publice, telefoanele mobile şi televiziunea prin cablu.

Cap. 3 discută modelul legătură de date şi protocoalele sale prin intermediul unui num ăr de exemple din ce în ce mai complexe. Se realizează, de asemenea, analiza acestor protocoale. După aceea, sunt discutate unele protocoale importante din lumea reală, printre care HDLC (folosit în reţelele de viteză scăzută şi medie) şi PPP (folosit în Internet).

Cap. 4 se referă la subnivelul de acces la mediu, care face parte din nivelul legătură de date. Problema fundamentală cu care se ocupă este cum să determine cine poate folosi reţeaua – atunci când reţeaua constă dintr-un singur canal partajat, aşa cum se întâmplă în majoritatea LAN-urilor şi în unele reţele de sateliţi. Sunt date multe exemple din domeniul LAN-urilor cu cablu sau fără (în special Ethernet), din cel al MAN-urilor fără fir, din cadrul reţelelor bazate pe Bluetooth şi al reţelelor de sateliţi. Tot aici sunt discutate şi punţile, care se folosesc pentru a interconecta LAN-urile.

Cap. 5 se ocupă de nivelul reţea, în special de dirijare, cu prezentarea mai multor algoritmi de dirijare, atât statici cât şi dinamici. Chiar dacă se folosesc algoritmi de rutare foarte buni, dacă traficul cerut este mai mare decât cel pe care îl poate dirija reţeaua, se ajunge la congestia reţelei, aşa că se va discuta despre congestie şi despre cum poate fi ea evitată. O variantă încă şi mai bună decât evitarea congestiei este oferirea unei garanţii de calitate a serviciilor. Şi acest subiect va fi abordat aici. Interconectarea reţelelor eterogene în inter-reţele conduce la numeroase probleme care sunt discutate aici. Se acordă mare atenţie nivelurilor din Internet .

Cap. 6 se ocupă de nivelul transport. Se discută pe larg protocoalele orientate pe conexiuni, deoarece ele sunt necesare în numeroase aplicaţii. Se discută în detaliu un exemplu de serviciu de transport şi implementarea sa. Este prezentat chiar şi codul sursă pentru acest exemplu simplu, pentru a se putea demonstra modul în care poate fi el implementat. Ambele protocoale din Internet – UDP şi TCP – sunt discutate în detaliu şi este abordată problema performanţelor lor. În plus, se discută despre problemele impuse de reţelele fără fir.

Cap. 7 se ocupă de nivelul aplicaţie, de protocoalele şi aplicaţiile sale. Primul subiect este DNS, care este cartea de telefoane a Internet-ului. Apoi urmează poşta electronică, inclusiv o discuţie despre protocoalele sale. Apoi ne vom muta atenţia asupra Web-ului, cu discuţii detaliate despre conţinut static, conţinut dinamic, ce se întâmplă la client, ce se întâmplă pe server, protocoale, performanţă, Web fără fir.

În cele din urmă vom examina informaţia multimedia care este transmisă prin reţea, inclusiv fluxuri audio, radio prin Internet şi video la cerere.

Cap. 8 se referă la securitatea reţelelor. Acest subiect include aspecte legate de fiecare dintre niveluri, aşa că este mai uşor de tratat către final, când toate nivelurile au fost deja explicate pe larg. Capitolul începe cu o introducere în criptografie. În continuare, este prezentat modul în care criptografia poate fi utilizată pentru a securiza comunicaţiilor, poşta electronică şi Web-ul. Cartea se încheie cu o discuţie despre anumite domenii în care securitatea interferează cu intimitatea, libertatea de exprimare, cenzura, precum şi alte probleme sociale care decurg de aici.

Cap. 9 conţine o listă adnotată de lecturi sugerate, aranjate în ordinea capitolelor. Lista este gândită ca un ajutor pentru cititorii care doresc să continue studiul reţelelor. Capitolul are de asemenea o bibliografie alfabetică a tuturor referinţelor citate în această carte.

Situl Web al autorului de la Prentice Hall: *http:llwww.prenhall.comltanenbaum* are o pagină cu legături la mai multe sinteze, liste de întrebări frecvente (FAQs), companii, consorţii industriale, organizaţii profesionale, organizaţii de standardizare, tehnologii, lucrări ştiinţifice şi altele.

1.9 REZUMAT

Reţelele de calculatoare pot fi utilizate pentru numeroase servicii, atât pentru firme cât şi pentru persoane particulare. Pentru companii, reţelele de calculatoare personale care folosesc servere partajate asigură accesul la informaţiile corporaţiei. De obicei, acestea urmează modelul client-server, cu staţiile de lucru clienţi pe mesele de lucru ale angajaţilor accesând serverele puternice din camera maşinilor. Pentru persoane particulare, reţelele oferă acces la o mulţime de informaţii şi de resurse de divertisment. De cele mai multe ori persoanele particulare accesează Internet-ul folosind un modem pentru a apela un ISP, deşi din ce în ce mai mulţi utilizatori au chiar şi acasă o conexiune Internet fixă, permanentă. Un domeniu care se dezvoltă rapid este acela al reţelelor fără fir, care conduc la dezvoltarea de noi aplicaţii, cum ar fi mobilitatea accesului la poşta electronică şi comerţul mobil.

În mare, reţelele pot fi împărţite în LAN-uri, MAN-uri, WAN-uri şi inter-reţele, fiecare cu caracteristicile, tehnologiile, vitezele şi rolurile sale proprii. LAN-urile acoperă suprafaţa unei clădiri şi lucrează la viteze mari, MAN-urile acoperă suprafaţa unui oraş – de exemplu reţeaua de televiziune prin cablu, care este actualmente folosită de mulţi dintre utilizatori şi pentru conectarea la Internet. WAN-urile se întind pe suprafaţa unei ţări sau a unui continent. LAN-urile şi MAN-urile sunt necomutate (adică nu au rutere); WAN-urile sunt comutate. Reţelele fără fir devin din ce în ce mai populare, în special la nivelul reţelelor locale. Reţelele pot fi interconectate pentru a forma inter-reţele.

Programele de reţea constau din protocoale, adică reguli prin care procesele pot să comunice. Protocoalele pot fi fie fără conexiuni, fie orientate pe conexiuni.

Majoritatea reţelelor asigură suport pentru ierarhiile de protocoale, fiecare nivel asigurând servicii pentru nivelurile de deasupra sa şi izolându-le de detaliile protocoalelor folosite în nivelurile de mai jos. Stivele de protocoale se bazează în mod tipic fie pe modelul OSI, fie pe modelul TCP/IP. Ambele modele posedă niveluri reţea, transport şi aplicaţie, dar ele diferă în ceea ce priveşte celelalte niveluri. Problemele care apar în procesul de proiectare a acestor protocoale includ multiplexarea, controlul traficului, controlul erorilor şi încă altele. O mare parte a acestei cărţi este dedicată protocoalelor şi proiectării lor.

Reţelele oferă servicii utilizatorilor lor. Aceste servicii pot fi orientate pe conexiune sau fără conexiune. În anumite reţele, serviciile fără conectare sunt oferite la un anumit nivel şi pot fi completate cu serviciile orientate pe conexiune oferite de un alt nivel.

Ca reţele bine-cunoscute sunt menţionate Internet-ul, reţelele ATM, Ethernet-ul şi LAN-ul fără fir, standard denumit IEEE 802.11. Internet-ul a evoluat din ARPANET, prin adăugarea de noi reţele pentru a se forma o inter-reţea. În prezent, Internet-ul este în fapt o colecţie de multe mii de reţele şi nu o singură reţea. Ceea ce caracterizează această colecţie este folosirea stivei TCP/IP peste tot. Reţelele ATM sunt răspândite mai ales în sistemele de telefonie pentru trafic de date intensiv. Ethernet-ul este cea mai populară reţea locală şi este implementată în majoritatea companiilor mari şi în universităţi. În fine, reţelele locale fără fir, cu viteze de transfer surprinzător de mari (până la 54 Mbps) încep să fie folosite pe scară largă.

Pentru a putea determina mai multe calculatoare să comunice între ele este nevoie de o importantă muncă de standardizare, atât pentru partea de echipamente (hardware), cât şi pentru partea de programe (software). Organizaţiile ca ITU-T, ISO, IEEE şi IAB administrează diverse părţi din procesul de standardizare.

1.10PROBLEME

1. Imaginaţi-vă că v-aţi dresat câinele St. Bernard, pe nume Bernie, ca, în locul clasicei sticle cu rom, să poarte o cutie cu trei benzi de 8 mm. (Când ţi se umple discul, respectiva cutie reprezintă o ur

genţă.) Aceste benzi conţin fiecare câte 7 gigabytes. Câinele poate călători până la dvs., oriunde v-aţi afla, cu 18 km/h. Pentru ce ordin de distanţe are Bernie o viteză mai mare de transmisie a datelor decât o linie a cărei viteză de transfer (fără supraîncărcare) este de 150 Mbps?

2. O alternativă la un LAN este pur şi simplu un mare sistem, cu divizarea timpului cu terminale pentru toţi utilizatorii. Prezentaţi două avantaje ale unui sistem client-server care foloseşte un LAN.

3. Performanţa unui sistem client-server este influenţată de doi factori ai reţelei: lărgimea de bandă (câţi biţi poate transporta într-o secundă) şi latenţa (câte secunde durează transferul primului bit de la client la server). Daţi un exemplu de reţea care are şi lărgime de bandă ridicată şi latenţă mare. Apoi daţi un exemplu de reţea cu lărgime de bandă scăzută şi latenţă mică.

4. Pe lângă lărgime de bandă şi latenţă, ce alt parametru este necesar pentru a caracteriza calitatea serviciilor oferite de o reţea folosită pentru trafic de voce digitizată?

5. Un factor de întârziere al unui sistem memorează-şi-retransmite cu comutare de pachete este cât de mult timp ia operaţia de stocare şi retrimitere a unui mesaj printr-un comutator. Dacă timpul de comutare este de 10 ps, este acesta un factor important în răspunsul unui sistem client-server în care clientul este în New York şi serverul în California? Presupuneţi că viteza de propagare a

6. semnalului printr-un fir de cupru sau prin fibra optică ar fi de 2/3 din viteza luminii în vid.

Un sistem client-server foloseşte o reţea-satelit, cu satelitul amplasat la o înălţime de 40.000 km. În cazul optim, care este întârzierea cu care vine răspunsul la o cerere?

7. În viitor, când toată lumea va avea acasă un terminal conectat la o reţea de calculatoare, vor deveni posibile referendumuri publice imediate pe subiecte de legislaţie importante. În ultimă instanţă ar putea fi chiar eliminate parlamentele, pentru a lăsa voinţa poporului să se exprime direct. Aspectele pozitive ale unei astfel de democraţii directe sunt destul de evidente; discutaţi unele din aspectele negative.

8. O colecţie de cinci rutere trebuie să fie conectată într-o subreţea punct-la-punct. Între două rutere proiectanţii pot instala o linie de mare viteză, o linie de viteză medie, o linie de viteză scăzută sau nici o linie. Dacă generarea şi examinarea fiecărei topologii pe calculator durează 100 ms, cât timp va dura examinarea tuturor topologiilor pentru a o găsi pe cea care se potriveşte cel mai bine cu încărcarea prevăzută?

9. Un grup de 2^n-l rutere sunt interconectate într-un arbore binar centralizat, cu un ruter în fiecare nod al arborelui. Ruterul *i* comunică cu ruterul *j* trimiţând un mesaj rădăcinii arborelui. Rădăcina trimite apoi mesajul înapoi în jos până la *j*.

Deduceţi o expresie aproximativă pentru numărul mediu de salturi pe mesaj în cazul unui număr *n* mare, presupunând că toate perechile de rutere sunt la fel de probabile.

10. Un dezavantaj al unei subreţele cu difuzare este risipa de capacitate datorată multiplelor gazde care încearcă să acceseze canalul în acelaşi timp. Ca un exemplu simplist, să presupunem că timpul este împărţit în intervale discrete şi fiecare din cele *n* gazde încearcă să utilizeze canalul cu probabilitatea *p* în timpul fiecărui interval. Ce fracţiune din intervale se pierde datorită coliziunilor?

11. Care sunt două din motivele utilizării protocoalelor organizate pe niveluri?

12. Preşedintelui Companiei de Vopsele Speciale îi vine ideea să lucreze împreună cu un producă

tor local de bere în scopul de a produce o cutie de bere invizibilă (ca o măsură anti-gunoi). Preşedintele comandă departamentului său juridic să analizeze ideea, iar acesta cere ajutorul, la rândul său, departamentului de ingineri. Ca rezultat, inginerul şef îl cheamă pe inginerul-şef de la cealaltă firmă pentru a discuta aspectele tehnice ale proiectului. Apoi, inginerii prezintă un raport către departamentele juridice respective, iar acestea aranjează prin telefon aspectele legale. În final, cei doi preşedinţi de firme discută partea financiară a afacerii. Este acesta un exemplu de protocol multinivel în sensul modelului OSI?**6.** Care sunt adresele

SAP în cazul difuzării radio FM ?

13. Care este principala diferenţă între comunicarea fără conexiuni şi comunicarea orientată pe conexiuni?

14. Două reţele furnizează, fiecare, servicii orientate pe conexiuni sigure. Una din ele oferă un flux sigur de octeţi, iar cealaltă oferă un flux sigur de mesaje. Sunt acestea identice? Dacă da, de ce se face această distincţie? Dacă nu, exemplificaţi prin ce diferă.

15. Ce înseamnă „negociere” atunci când se discută protocoalele de reţea? Daţi un exemplu.

16. În fig. l-19 este prezentat un serviciu. Există şi servicii implicite în această figură? Dacă da, unde? Dacă nu, de ce nu?

17. În unele reţele, nivelul legătură de date tratează erorile de transmisie, solicitând retransmiterea cadrelor deteriorate. Dacă probabilitatea de a se strica un cadru este *p,* care este numărul mediu de transmisii necesare pentru a trimite un cadru, în cazul în care confirmările nu se pierd niciodată?

18. Care dintre nivelurile OSI se ocupă de fiecare din următoarele sarcini:

a) Descompunerea fluxului de biţi transmişi în cadre.
b) Determinarea traseului care trebuie folosit în subreţea.

c) TDPU-rile încapsulează pachete sau invers? Discuţie.

19. Dacă unităţile de date schimbate la nivelul legătură de date se numesc cadre şi unităţile de date schimbate la nivelul reţea se numesc pachete, pachetele încapsulează cadre sau cadrele încapsulează pachete? Explicaţi răspunsul dat.

20. Un sistem are o ierarhie de protocoale organizată pe n niveluri. Aplicaţiile generează mesaje de lungime M octeţi. La fiecare nivel este adăugat un antet de h octeţi. Ce fracţiune din lăţimea benzii reţelei este ocupată de antete?

21. Prezentaţi două aspecte comune modelului de referinţă OSI şi modelului de referinţă TCP/IP. Prezentaţi apoi două aspecte prin care modelele diferă.

22. Care este principala deosebire între TCP şi UDP?

23. Subreţeaua din fig. l-25(b) a fost proiectată pentru a putea rezista unui război nuclear. Câte bombe ar fi necesare pentru a partiţiona nodurile sale în două seturi complet deconectate? Presupuneţi că orice bombă distruge un nod şi toate legăturile conectate cu el.

24. Internet-ul îşi dublează dimensiunea o dată la aproximativ 18 luni. Deşi nimeni nu ştie cu siguranţă, se estimează numărul gazdelor la 100 de milioane în 2001. Folosiţi aceste date pentru a calcula numărul de gazde Internet prevăzut pentru anul 2010. Puteţi crede acest scenariu? Explicaţi de ce da sau de ce nu.

25. La transferul unui fişier între două calculatoare există (cel puţin) două strategii de confirmare. Conform primei strategii, fişierul este descompus în pachete care sunt confirmate individual de către server, dar transferul de fişiere pe ansamblu nu este confirmat. În a doua strategie, pachetele nu sunt confirmate individual, dar la sfârşit este confirmat întregul fişier. Discutaţi aceste două abordări.

26. De ce foloseşte ATM-ul celule mici, de lungime fixă?

27. Cât de lung era un bit în standardul original 802.3 măsurat în metri? Folosiţi viteza de transmisie de 10 Mbps şi presupuneţi că viteza de transmisie prin cablu coaxial este de 2/3 din viteza de propagare a luminii în vid.

28. O imagine are 1024 x 768 pixeli şi reţine câte 3 octeţi pentru fiecare pixel. Presupuneţi că imaginea este necomprimată. Cât durează transmisia ei pe un canal de modem de 56 Kbps ? Dar printr-un modem de cablu de 1 Mbps? Dar prin Ethernet la 10 Mbps? Dar prin Ethernet la 100 Mbps ?

29. Ethernet-ul şi reţelele fără fir au unele asemănări şi deosebiri. O proprietate a Ethernet-ului este aceea că un singur cadru poate fi transmis la un moment dat pe mediu. Are şi 802.11 această proprietate? Discutaţi răspunsul dat.

30. Reţelele fără fir sunt uşor de instalat, ceea ce le face mai ieftine, deoarece de cele mai mult ori operaţia de instalare depăşeşte semnificativ costul echipamentelor. Totuşi, aceste reţele au şi unele dezavantaje. Numiţi două dintre ele.

31. Prezentaţi două avantaje şi două dezavantaje ale existenţei standardelor internaţionale pentru protocoalele de reţea.

32. Atunci când un sistem dispune de o parte permanentă şi de o parte detaşabilă, de exemplu un cititor de CD-uri şi un CD-ROM, este important ca sistemul să fie standardizat, astfel ca diferite firme să poată realiza atât părţile permanente cât şi cele mobile şi ca ele să se potrivească fără probleme. Daţi trei exemple din afara industriei de calculatoare unde există astfel de standarde internaţionale. Indicaţi apoi trei domenii din afara industriei de calculatoare unde nu există astfel de standarde.

33. Alcătuiţi o listă de activităţi pe care le faceţi zilnic şi în care sunt implicate reţele de calculatoare. Cum ar fi viaţa voastră alterată dacă aceste reţele ar fi deconectate la un moment dat ?

34. Descoperiţi ce reţele sunt utilizate în şcoala sau la locul de muncă. Descrieţi tipurile de reţele, topologiile şi metodele de comutare folosite acolo.

35. Programul *ping* vă permite să trimiteţi un pachet de test la o locaţie dată pentru a vedea cât de mult durează până când acesta ajunge acolo şi înapoi. Încercaţi să folosiţi *ping* pentru a vedea cât de mult durează transferul pachetului între locul în care vă găsiţi şi alte câteva locuri cunoscute. Din aceste date, calculaţi timpul de tranzit într-o sigură direcţie în funcţie de distanţă. Este bine să folosiţi universităţile deoarece locaţiile serverelor lor sunt cunoscute foarte bine. De exemplu, *berkley.edu* este în Berkley, California, *mit.edu* este în Cambridge, Massachusetts, *vu.nl* este în Amsterdam, Olanda, *www.usyd.edu.au* este în Sydney, Australia şi *www.uct.ac.za* este în Cape Town, Africa de Sud.

36. Vizitaţi situl Web al IETF, *www.ietf.org* pentru a vedea ce mai fac. Alegeţi un proiect care vă place şi scrieţi un raport de jumătate de pagină despre problemă şi despre o soluţie propusă.

37. Standardizarea este foarte importantă în lumea reţelelor. ITU şi ISO sunt principalele organizaţii oficiale de standardizare. Vizitaţi siturile lor Web, *www.itu.org* şi *www.iso.org*, respectiv, şi aflaţi despre munca lor de standardizare. Scrieţi un scurt raport despre tipurile de lucruri pe care le-au standardizat.

38. Internet-ul este alcătuit dintr-un mare număr de reţele. Aranjarea lor determină topologia În- ternet-ului. O importantă cantitate de informaţii despre topologia Internet-ului este disponibilă online. Folosiţi un motor de căutare pentru a afla mai multe despre acest subiect şi scrieţi un scurt raport care să rezume informaţiile pe care le-aţi găsit.

NIVELUL FIZIC

În continuare vom analiza trei tipuri de medii de transmisie: ghidate (cablu din cupru şi fibre optice), fără fir (unde radio terestre) şi prin satelit. Acest material furnizează informaţiile fundamentale referitoare la tehnologiile de comunicaţie folosite în reţelele moderne.

Restul capitolului este dedicat descrierii a trei exemple de sisteme de comunicaţie folosite în practică pentru reţele cu răspândire geografică largă. Vom începe cu sistemul telefonic, studiind trei variante: sistemul de telefonie fixă, sistemul de telefonie mobilă şi sistemul bazat pe cablu de televiziune. Toate acestea folosesc fibra optică pentru implementarea coloanei vertebrale, dar sunt organizate diferit şi folosesc tehnologii diferite pentru ultima milă a legăturii.

2.1 BAZELE TEORETICE ALE COMUNICĂRII DE DATE

Informaţia poate fi transmisă prin cablu folosind variaţia unor proprietăţi fizice ale semnalului cum ar fi tensiunea şi intensitatea curentului. Reprezentând valoarea tensiunii sau a intensităţii curentului ca o funcţie de timp, *f(t),* putem modela comportamentul semnalului şi îl putem analiza matematic. Această analiză face

subiectul următoarelor secţiuni.

2.1.1 Analiza Fourier

La începutul secolului XIX, matematicianul francez Jean-Baptiste Fourier a demonstrat că orice funcţie *g(t),* cu evoluţie rezonabilă şi periodică cu perioada T, poate fi construită prin însumarea unui număr (posibil infinit) de sinusoide şi cosinusoide:

$$g(t) = \frac{1}{2}c + \sum_{n=1}^{\infty} a_n \sin(2\pi nft) + \sum_{n=1}^{\infty} b_n \cos(2\pi nft) \qquad (2\text{-}1)$$

unde $f = 1 / T$ este frecvenţa fundamentală, iar a_n şi b_n sunt amplitudinile sinusoidelor şi cosinusoide- lor **armonicei** (termenului) de ordinul n, iar c este o constantă. Această descompunere este numită **serie Fourier**. Pornind de la seria Fourier, funcţia poate fi reconstruită; aceasta înseamnă că, dacă perioada T este cunoscută şi amplitudinile sunt date, funcţia de timp originală poate fi obţinută prin evaluarea sumelor din ecuaţia 2-1.

Un semnal de durată finită (proprietate pe care o au toate semnalele) poate fi tratat presupunându-se că el repetă un anumit tipar la infinit (de exemplu, semnalul este acelaşi în intervalul de la T la $2T$ ca în intervalul de la 0 la T, etc.).

Amplitudinile a_n pot fi calculate pentru orice $g(t)$ dat prin multiplicarea ambilor membri ai ecuaţiei 2-l cu $\sin(2nkfi)$ urmată de integrarea de la 0 la T. Deoarece

$$\int_0^T \sin(2\pi kft)\sin(2\pi nft)\,dt = \begin{cases} 0 & \text{pentru } k \neq n \\ T/2 & \text{pentru } k = n \end{cases}$$

numai un singur termen al sumei nu se anulează: a_n . Suma de cosinusuri – cea cu b_n – se anulează complet. Similar, multiplicând membrii ecuaţiei 2-l cu $\cos(2nkfi)$ şi integrând de la 0 la T, putem obţine b_n. Prin integrarea ambilor membri ai ecuaţiei originale, se poate obţine c. Rezultatele obţinute prin efectuarea acestor operaţii sunt:

$$a_n = \frac{2}{T}\int_0^T s(t)\sin(2\pi nft)\,dt \qquad b_n = \frac{2}{T}\int_0^T g(t)\cos(2\pi nft)\,dt \qquad c = \frac{2}{T}\int_0^T g(t)\,dt$$

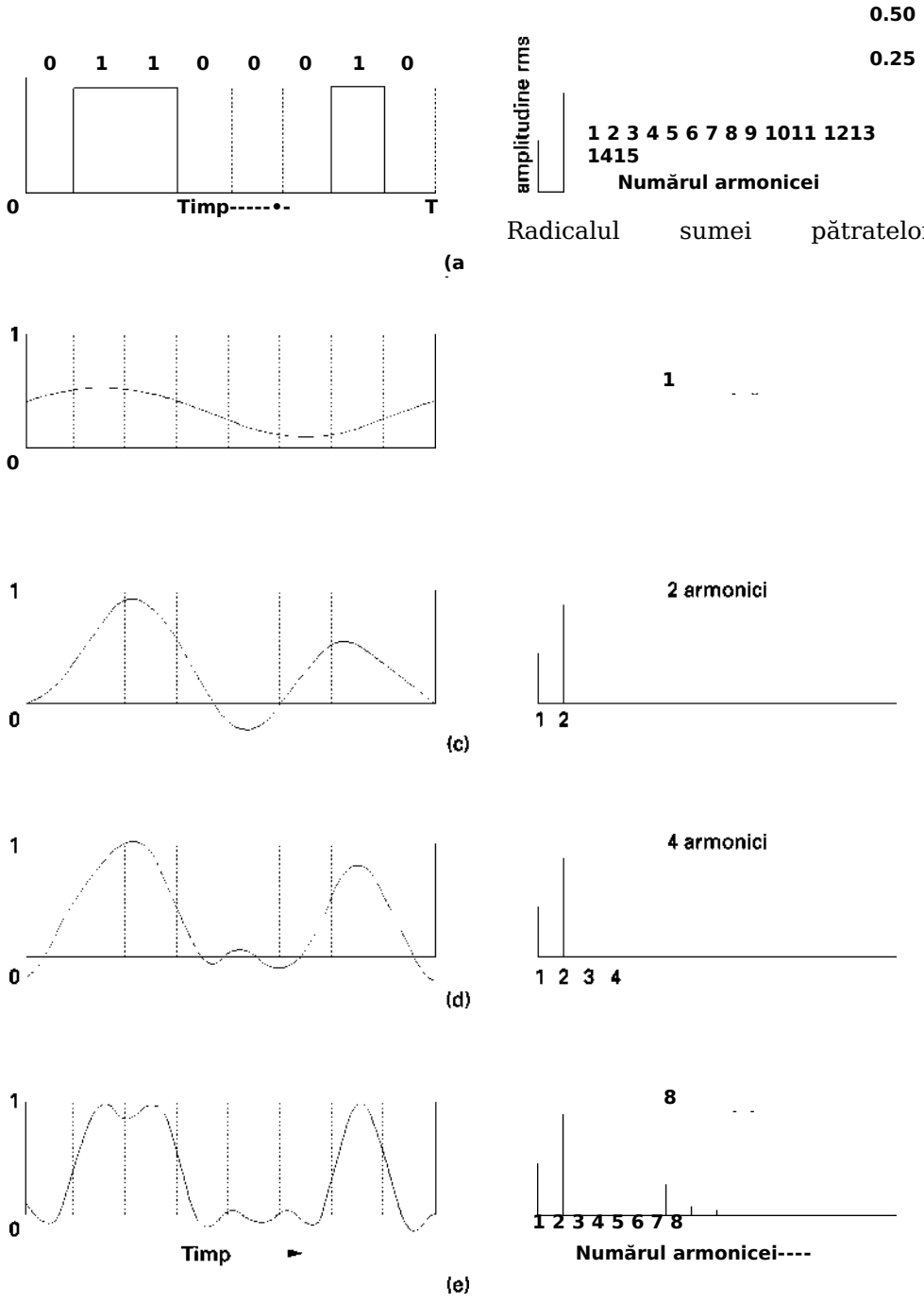
2.1.2 Semnalele cu bandă de frecvenţă limitată

Pentru a face legătura dintre cele prezentate şi comunicaţia de date să considerăm următorul exemplu: transmisia caracterului ASCII „b” codificat pe un octet. Biţii care urmează a fi transmişi sunt 01100010. Partea din stânga a fig. 2-1(a) reprezintă tensiunea la ieşire emisă de calculatorul care transmite. Din analiza Fourier a acestui semnal rezultă următorii coeficienţi:

$$a_n = \frac{}{n\pi}[\cos(n\pi / 4) - \cos(3n\pi / 4) + \cos(6n\pi / 4) - \cos(7n\pi / 4)]$$

$$b_n = \frac{}{}[\sin(3n\pi / 4) - \sin(n\pi / 4) + \sin(7n\pi / 4) - \sin(6n\pi / 4)]$$

$$c = 3/4$$



Radicalul sumei pătratelor

Fig. 2-1. (a) Un semnal binar şi radicalul sumei pătratelor amplitudinilor Fourier. (b) – (e) Aproximaţii succesive

amplitudinilor, $\hat{}a_t + b_2$, pentru primii termeni este prezentat în partea dreaptă a fig. 2-1(a). Aceste valori sunt cele care ne interesează, deoarece pătratele lor sunt

proporţionale cu energia transmisă la frecvenţa respectivă.

Nu există un mijloc de transmisie care să poată trimite semnale fără pierdere de putere în timpul procesului. Dacă toate componentele Fourier ar fi micşorate în aceeaşi măsură, atunci semnalul rezultat ar fi atenuat în amplitudine, dar nu ar prezenta distorsiuni [ar avea aceeaşi formă ca cea din fig. 2-1(a)]. Din păcate, orice mijloc de transmisie atenuează componente Fourier diferite cu factori diferiţi, introducând astfel distorsiuni. De obicei, amplitudinile sunt transmise fără atenuări de la 0 la o anumită frecvenţă f_p [măsurată în cicluri/secundă sau în Hertzi (Hz)] şi toate celelalte componente cu frecvenţe mai mari decât această frecvenţă de tăiere sunt puternic atenuate. Intervalul de frecvenţe transmise fără a fi atenuate semnificativ se numeşte **lărgime de bandă**. În practică, tăierea nu este verticală (şi deci frecvenţa de tăiere nu este exactă), astfel încât deseori lărgimea de bandă este aproximată ca intervalul dintre 0 şi frecvenţa de trecere pentru jumătate din puterea maximă.

Lărgimea de bandă este o proprietate fizică a mediului de transmisie şi de obicei depinde de construcţia, grosimea şi lungimea mediului. În unele cazuri, în circuit este introdus un filtru pentru a limita lărgimea de bandă disponibilă pentru fiecare client. De exemplu, un fir de telefon poate avea lărgimea de bandă de 1 MHz pentru distanţe scurte, dar companiile telefonice adaugă un filtru ce limitează fiecare client la aproximativ 3100 Hz. Această variantă este adecvată pentru vorbire inteligibilă şi îmbunătăţeşte eficienţa sistemului prin limitarea utilizării de resurse de către clienţi.

Să vedem cum va arăta semnalul transmis dacă banda de frecvenţă folosită ar fi atât de îngustă, încât numai frecvenţele foarte joase pot fi transmise [funcţia ar fi aproximată doar cu primii câţiva termeni ai ecuaţiei (2-1)]. Fig. 2-1(b) reprezintă semnalul rezultat dintr-un canal care permite numai primei armonici *(f,* fundamentală) să fie transmisă. Similar, fig. 2-1(c)-(e) prezintă spectrele şi funcţiile reconstruite pentru canale cu lărgime de bandă mai mare.

Fiind dată o rată de transmisie a biţilor de b biţi/secundă, timpul necesar pentru a transmite 8 biţi (de exemplu) este de $8/b$ secunde, frecvenţa primei armonice fiind $b/8$ Hz. O linie telefonică obişnuită, deseori numită **linie în bandă vocală** (**voice-grade line**), este limitată artificial de o frecvenţă de tăiere puţin peste 3000 Hz. Această restricţie impune ca numărul celei mai mari armonice care poate fi transmisă este aproximativ de $3000/(b/8)$, adică $24000/b$ (frecvenţa de prag nu este foarte exactă).

Bps	T (msec)	Prima armonică (Hz)	nr. armonice transmise
300	26.27	37.5	80
600	13.33	75	40
1200	6.67	150	20
2400	3.33	300	10
4800	1.67	600	5
9600	0.83	1200	2
19200	0.42	2400	1
38400	0.21	4800	0

Fig. 2-2. Relaţia între viteza de transfer a datelor şi armonice.

În fig. 2-2 sunt prezentate valorile pentru anumite viteze de transfer de date. Pornind de la aceste valori, este clar că încercarea de a transmite date la o viteză de 9600 bps folosind o linie telefonică obişnuită va transforma semnalul din fig. 2-1(a) în ceva asemănător cu fig. 2-1(c), fiind dificilă obţinerea secvenţei de biţi originale. Este evident că la viteze de transfer mai mari decât 38.4 Kbps nu există nici o speranţă de a recupera semnalele binare, chiar dacă mediul de transmisie ar fi lipsit în totalitate de zgomote. Cu alte cuvinte, limitând lărgimea de bandă se limitează şi viteza de transfer chiar şi pentru canalele perfecte. Oricum, există tehnici de codificare sofisticate, care folosesc mai multe niveluri de tensiune şi care pot atinge rate de transfer mai mari. Vom discuta aceste tehnici mai târziu în acest capitol.

2.1.3 Viteza maximă de transfer de date a unui canal

Încă din 1924, un inginer AT&T, H. Nyquist a descoperit că şi un canal perfect are o capacitate limitată de transmisie. El a dedus o ecuaţie care exprimă viteza maximă de transfer de date pentru un canal fără zgomote, cu lărgime de bandă finită. În 1948, Claude Shannon a continuat cercetările lui Nyquist exprimând această limită pentru un canal supus zgomotului aleatoriu (termodinamic) (Shannon 1948). Noi nu vom face aici decât o scurtă prezentare a acestor rezultate, acum devenite clasice.

Nyquist a demonstrat că dacă un semnal arbitrar este transmis printr-un filtru de frecvenţe joase cu lărgime de bandă *H,* semnalul filtrat poate fi complet reconstruit prin efectuarea a numai *2H* eşantioane pe secundă. Eşantionarea semnalului la o viteză mai mare decât 2*H*/secundă este inutilă, deoarece componentele cu o frecvenţă mai înaltă pe care aceste eşantioane le-ar putea obţine au fost deja filtrate. Dacă semnalul are *V* niveluri discrete, teorema lui Nyquist afirmă:

$$\text{viteza maximă de transfer de date} = 2H \log_2 V \text{ biţi / sec}$$

De exemplu, un canal de 3kHz, fără zgomote, nu poate transmite semnale binare (pe două niveluri) la o viteză mai mare de 6000 bps.

Până acum am studiat doar cazul canalelor fără zgomote. Dacă sunt prezente zgomote aleatoare, situaţia se deteriorează rapid. Iar un zgomot aleator (termic) datorat mişcării moleculelor în sistem va fi prezent întotdeauna. Dimensiunea zgomotului termic prezent se măsoară prin raportul dintre puterea semnalului şi puterea zgomotului, fiind numită **raportul semnal-zgomot**. Dacă notăm puterea semnalului cu *S* şi puterea zgomotului cu *N,* atunci raportul semnal-zgomot este *S/N.* De obicei, acest raport nu este specificat; în schimb, este dată expresia 10 $\log_{10} S/N$. Aceste unităţi sunt numite **decibeli** (dB). Un raport *S/N* egal cu 10 este de 10 dB, un raport egal cu 100 este de 20 dB, un raport egal cu 1000 este de 30 dB şi aşa mai departe. De multe ori fabricanţii de amplificatoare stereo caracterizează banda de frecvenţă (domeniul de frecvenţă) în care produsul lor este liniar furnizând frecvenţele la care semnalul se atenuează cu 3 dB la fiecare capăt. Acestea sunt punctele în care factorul de amplificare este aproximativ înjumătăţită (deoarece $\log_{10} 3 \approx 0.5$).

Rezultatul cel mai important obţinut de Shannon este expresia pentru viteza maximă de transfer de date a unui canal cu zgomote, având lărgimea de bandă de *H*

Hz şi a cărui raport semnal-zgomot S/N este dat de:

$$\text{numărul maxim de biţi/sec} = H \log_2 (1+S/N)$$

De exemplu, un canal cu o bandă de frecvenţă de 3000 Hz şi zgomot termic de 30 dB (parametri tipici părţii analogice a sistemului telefonic) nu va putea transmite mult mai mult de 30.000 bps, indiferent de cât de multe sau de puţine niveluri are semnalul sau cât de multe sau puţine eşantioane sunt luate. Rezultatele lui Shannon au fost obţinute folosind atât argumente teoretice cât şi argumente informaţionale şi se aplică oricărui canal supus zgomotelor termice. Contraexemplele ar trebui plasate în aceeaşi categorie cu maşinile perpetuum mobile. Ar trebui remarcat şi că această viteză nu este decât o limitare superioară pe care sistemele reale o ating rareori.

2.2 MEDII DE TRANSMISIE GHIDATĂ

Scopul nivelului fizic este de a transporta o secvenţă de biţi de la o maşină la alta. Pentru transmisia efectivă pot fi utilizate diverse medii fizice. Fiecare dintre ele este definit de lărgime proprie de bandă, întârziere, cost, dar şi de uşurinţa de instalare şi întreţinere. Aceste medii pot fi împărţite în două grupe mari: medii ghidate, cum sunt cablul de cupru şi fibrele optice şi medii neghidate, cum sunt undele radio şi laserul. Vom arunca o privire asupra acestora în următoarele secţiuni.

2.2.1 Medii magnetice

Una din cele mai obişnuite metode de a transporta date de la un calculator la altul este să se scrie datele pe o bandă magnetică sau pe un suport reutilizabil (de exemplu, DVD-uri pentru înregistrare), să se transporte fizic banda sau discul la maşina de destinaţie, după care să se citească din nou datele. Cu toate că această metodă nu este la fel de sofisticată precum folosirea unui satelit de comunicaţie geosincron, ea este de multe ori mai eficientă din punct de vedere al costului, mai ales pentru aplicaţiile în care lărgimea de bandă sau costul pe bit transportat sunt factori cheie.

Un calcul simplu va confirma acest punct de vedere. O bandă Ultrium standard industrial poate înmagazina 200 gigaocteţi. Într-o cutie cu dimensiunile 60 x 60 x 60 cm pot să încapă cam 1000 de astfel de benzi, ceea ce înseamnă o capacitate totală de 200 de terraocteţi sau 1600 terrabiţi (1.6 petabiţi). O cutie cu benzi poate fi distribuită oriunde în Statele Unite în 24 de ore de către Federal Express sau de alte companii. Banda de frecvenţă efectivă a acestei transmisii este de 1600 terrabiţi / 84600 secunde, adică 19Gbps. Dacă destinaţia ar fi la distanţă de numai o oră cu maşina, lărgimea de bandă s-ar mări la peste 400Gbps. Nici o reţea de calculatoare nu poate să se apropie de o asemenea viteză.

Pentru o bancă în care datele sunt de ordinul gigaocteţilor şi trebuie salvate zilnic pe o altă maşină (pentru ca banca să poată funcţiona în continuare chiar şi în

urma unor inundaţii puternice sau unui cutremur), probabil că nici o altă tehnologie de transmisie nu e comparabilă cu performanţa atinsă de banda magnetică. Desigur, reţelele devin din ce în ce mai rapide, dar şi capacităţile benzilor magnetice cresc.

Dacă ne uităm la cost, vom obţine aceeaşi situaţie. Atunci, dacă sunt cumpărate en-gros, benzile Ultrium ajung să coste în jur de 40 de dolari pe bucată. O bandă poate fi refolosită de cel puţin 10 ori, astfel încât costul benzii este aproape de 4000 de dolari/cutie/utilizare. Dacă adăugăm încă 1000 de dolari pentru transport (probabil mult mai ieftin), vom avea un cost de 5000 de dolari pentru a transporta 200 de terraocteţi. De aici rezultă că un gigaoctet poate fi transportat la un preţ mai mic de 3 cenţi. Nici o reţea nu poate concura cu un astfel de preţ. Morala poveştii:

Niciodată nu subestima lărgimea de bandă a unui camion încărcat cu benzi magnetice care goneşte la vale pe autostradă.

2.2.2 Cablul torsadat

Deşi caracteristicile de lărgime de bandă ale mediilor magnetice sunt excelente, performanţele legate de întârzieri sunt slabe. Timpul de transmisie nu se măsoară în milisecunde, ci în minute sau ore. Pentru multe aplicaţii este nevoie de o conexiune on-line. Unul dintre cele mai vechi medii de transmisie, rămas cel mai utilizat mediu, este **cablul torsadat**. O pereche torsadată este formată din două fire de cupru izolate, fiecare având o grosime tipică de 1 mm. Firele sunt împletite într-o formă elicoidală, ca o moleculă de ADN. Împletirea se face pentru că două fire paralele constituie o bună antenă. Dacă firele sunt împletite, undele din diferite împletiri se anulează, astfel încât radiaţia firului este scăzută eficient.

Cea mai cunoscută aplicaţie a cablului torsadat este sistemul telefonic. Aproape toate telefoanele sunt conectate la centrala telefonică printr-un cablu torsadat. Cablurile torsadate se pot întinde pe mai mulţi kilometri fără amplificare, dar pentru distanţe mai mari, sunt necesare repetoare. Atunci când mai multe cabluri torsadate sunt grupate în paralel – cum sunt de exemplu toate firele de la un bloc de locuinţe legate la centrala telefonică – ele sunt legate împreună şi încapsulate într-un material protector. Dacă perechile de fire nu ar fi fost împletite, cablurile grupate astfel împreună ar fi interferat. În anumite părţi ale lumii, unde liniile telefonice sunt montate pe stâlpi, sunt des întâlnite cablurile cu diametrul de câţiva centimetri.

Cablurile torsadate pot fi folosite atât pentru transmisia semnalelor analogice cât şi pentru transmisia de semnale digitale. Lărgimea de bandă depinde de grosimea firului şi de distanţa parcursă, dar, în multe cazuri, se poate atinge o viteză de mai mulţi megabiţi pe secundă pe distanţe de ordinul a câţiva kilometri. Datorită performanţei satisfăcătoare şi a costului scăzut, cablurile torsadate sunt foarte larg folosite în prezent şi probabil că vor rămâne larg folosite şi în următorii ani.

Există numeroase tipuri de cabluri torsadat, două dintre acestea fiind importante pentru reţelele de calculatoare. Perechile torsadate din **Categoria 3** sunt formate din două fire izolate răsucite unul în jurul celuilalt cu pas mare. De obicei, patru astfel de perechi sunt grupate într-un material plastic, pentru a le proteja şi pentru a le ţine împreună. Până în 1988, cele mai multe clădiri cu birouri aveau un cablu de categoria 3, care pornea din panoul central de la fiecare etaj către fiecare birou. Această schemă permitea ca maxim patru telefoane obişnuite, sau maxim două

telefoane cu mai multe linii, toate aflate în acelaşi birou, să poată fi cuplate la centrala telefonică prin panoul central.

Începând din 1988, au fost introduse cablurile de **Categoria 5**, mai performante. Ele sunt similare celor din categoria 3, dar au mai multe răsuciri pe centimetru (pas de răsucire mai mic), rezultând o interferenţă (diafonie) scăzută şi o mai bună calitate a semnalului pe distanţe mari, ceea ce le face mai adecvate comunicaţiilor la viteze mari între calculatoare. Categoriile mai noi sunt 6 şi 7, care sunt capabile să trateze semnale cu banda de frecvenţă de 250 MHz şi, respectiv, 600 MHz (faţă de numai 16MHz sau 100MHz pentru categoriile 3 şi, respectiv, 5) .

Pentru a le deosebi de cablurile torsadate voluminoase, ecranate şi scumpe, pe care IBM le-a introdus la începutul anilor ‘80, dar care nu au devenit populare în afara instalaţiilor IBM, aceste tipuri de cabluri sunt cunoscute sub numele de cabluri UTP (Unshielded Twisted Pair, rom: cablu torsadat neecranat). Torsadarea firelor este ilustrată în fig. 2-3.

Fig. 2-3. (a) Cablu UTP cat. 3. (b) Cablu UTP cat. 5.

2.2.3 Cablu Coaxial

Un alt mediu uzual de transmisie este cablul coaxial (cunosc ut printre utilizatorii săi sub numele de „coax” şi este pronunţat “co-ax”). El are o ecranare mai bună decât cablurile torsadate, putând acoperi distanţe mai mari la rate de transfer mai mari. Există două tipuri de cabluri coaxiale folosite pe scară largă. Primul, cablul de 50 de ohmi, este folosit frecvent când se doreşte transmisie digitală de la început. Al doilea tip, cablul de 75 de ohmi, este frecvent folosit în transmisia analogică şi televiziunea prin cablu, dar devine tot mai important o data cu apariţia Internetului prin cablu. Această clasificare are la bază un criteriu stabilit mai mult pe considerente istorice decât pe considerente tehnice (de exemplu, primele antene dipol aveau o impedanţă de 300 de ohmi şi existau transformatoare de impedanţă 4 : 1, care erau uşor de folosit).

Un cablu coaxial este format dintr-o sârmă de cupru rigidă, protejată de un material izolator. Acest material este încapsulat într-un conductor circular, de obicei sub forma unei plase strâns întreţesute. Conductorul exterior este acoperit cu un înveliş de plastic protector. În fig. 2-4 este prezentată o vedere în secţiune a cablului coaxial.

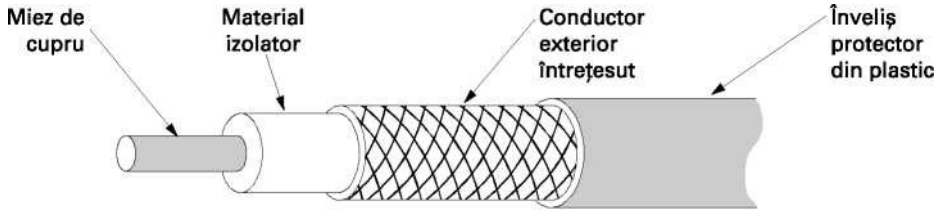


Fig. 2-4. Un cablu coaxial.

Structura şi ecranarea cablului coaxial asigură o bună împletire a necesităţilor semnificative de lărgime de bandă şi totodată de imunitate excelentă la zgomot. Lărgimea de bandă poate depinde de calitatea cablului, de lungime, şi de raportul semnal-zgomot al semnalului de date. Cablurile moderne au o bandă de frecvenţă de aproape 1 GHz. Cablurile coaxiale erau folosite pe scară largă în sistemul telefonic pentru linii întinse pe distanţe mari, dar au fost în mare parte înlocuite cu fibre optice. Oricum, cablul coaxial este utilizat în continuare în televiziunea prin cablu şi în unele reţele locale.

2.2.4 Fibre optice

Mulţi dintre cei implicaţi în industria calculatoarelor sunt foarte mândri de viteza de evoluţie a tehnologiei calculatoarelor. Originalul (1981) IBM PC rula la o frecvenţă de ceas de 4,77 MHz. Douăzeci de ani mai târziu, calculatoarele personale pot rula la 2 GHz, ceea ce reprezintă o creştere a frecvenţei de 20 de ori pentru fiecare deceniu. Nu e rău deloc.

În aceeaşi perioadă, comunicaţiile de date pe arii întinse au evoluat de la o viteză de 56 Kbps (ARPANET) până la 1 Gbps (comunicaţiile optice moderne), o creştere de mai bine de 125 de ori pentru fiecare deceniu. În aceeaşi perioadă, frecvenţa erorilor a scăzut de la 10^{-5} per bit până aproape de zero.

Mai mult, procesoarele se apropie de limitele lor fizice, date de viteza luminii şi de problemele de disipare a căldurii. Din contră, folosind tehnologiile *actuale* de fibre optice, lărgimea de bandă care poate fi atinsă este cu siguranţă mai mare decât 50,000 Gbps (50 Tbps) şi sunt încă mulţi cei care caută materiale şi tehnologii mai performante. Limitarea practică actuală la aproximativ 10 Gbps este o consecinţă a imposibilităţii de a converti mai rapid semnalele electrice în semnale optice, deşi, în laborator, 100 Gbps a fost atinsă într-o singură fibră.

În cursa dintre calculatoare şi comunicaţii, acestea din urmă au învins. Implicaţiile complete ale lărgimii de bandă infinite (deşi nu la un cost nul) nu au fost încă abordate de o generaţie de oameni de ştiinţă şi ingineri învăţaţi să gândească în termenii limitărilor calculate de Nyquist şi Shannon, limitări stricte impuse de firele de cupru. Noua paradigmă convenţională spune că maşinile de calcul sunt extrem de lente, astfel că reţelele trebuie să evite cu orice preţ calculele, indiferent de lărgimea de bandă risipită. În această secţiune vom studia fibrele optice pentru a ne familiariza cu această tehnologie de transmisie.

Un sistem de transmisie optică este format din trei componente: sursa de lumină, mediul de transmisie şi detectorul. Prin convenţie, un impuls de lumină înseamnă un bit cu valoarea 1, iar absenţa luminii indică un bit cu valoarea 0. Mediul de transmisie este o fibră foarte subţire de sticlă. Atunci când interceptează un impuls luminos, detectorul generează un impuls electric. Prin ataşarea unei surse de lumină la un capăt al fibrei optice şi a unui detector la celălalt, obţinem un sistem unidirecţional de transmisie a datelor care primeşte un semnal electric, îl converteşte şi îl transmite ca impulsuri luminoase şi apoi reconverteşte ieşirea în semnale electrice la recepţie.

Acest sistem de transmisie ar fi pierdut din semnalele luminoase şi ar fi fost lipsit de importanţă în practică, dacă nu s-ar fi folosit un principiu interesant al fizicii:

când o rază luminoasă trece de la un mediu la altul, de exemplu de la siliciu la aer, raza este refractată (frântă) la suprafaţa de separaţie siliciu / aer ca în fig. 2-5. Se observă o rază de lumină incidentă pe suprafaţa de separaţie la un unghi a_1 care se refractă la un unghi p_1. Mărimea refracţiei depinde de proprietăţile celor două medii (în particular, de indicii lor de refracţie). Pentru unghiuri de incidenţă mai mari decât o anumită valoare critică, lumina este refractată înapoi în siliciu fără nici o pierdere. Astfel o rază de lumină, la un unghi egal sau mai mare decât unghiul critic, este încapsulată în interiorul fibrei, ca în fig. 2-5(b) şi se poate propaga pe mulţi kilometri, aparent fără pierderi.

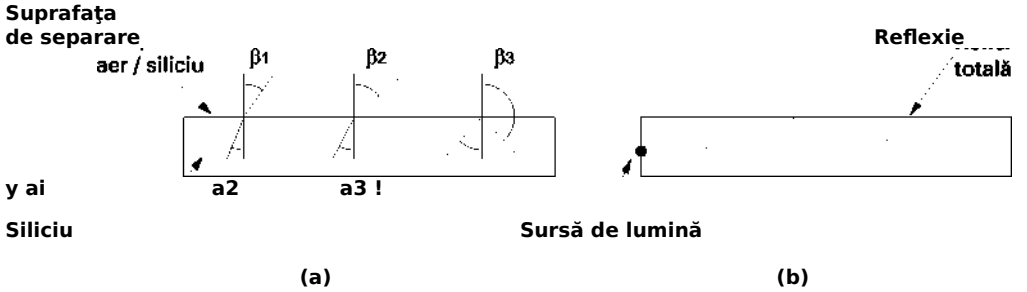


Fig. 2-5. (a) Trei exemple de raze de lumină în interiorul unei fibre de siliciu care cad pe suprafaţa de separaţie aer/siliciu la unghiuri diferite. (b) Încapsularea luminii prin reflexie totală.

În fig. 2-5(b) se poate observa o singura rază încapsulată, dar se pot transmite mai multe raze cu unghiuri de incidenţă diferite, datorită faptului că orice rază de lumină cu unghi de incidenţă la suprafaţa de separaţie mai mare decât unghiul critic va fi reflectată total. Se spune că fiecare rază are un **mod** diferit, iar fibra care are această proprietate se numeşte **fibră multi-mod**.

Oricum, dacă diametrul fibrei este redus la câteva lungimi de undă ale luminii, fibra acţionează ca un ghid de undă şi lumina se va propaga în linie dreaptă, fără reflexii, rezultând o **fibră monomod**. Aceste fibre sunt mai scumpe, dar sunt des folosite pentru distanţe mai mari. Fibrele mono- mod curente pot transmite date la 50 Gbps pe distanţe de 100 Km fără amplificare. În condiţii de laborator şi pentru distanţe mai mici s-au obţinut rate de transfer chiar şi mai mari.

Transmisia luminii prin fibre

Fibrele optice sunt fabricate din sticlă, iar sticla este fabricată la rândul ei din nisip, un material brut necostisitor, care se găseşte în cantităţi nelimitate. Producerea sticlei era cunoscută de egiptenii din Antichitate, dar pentru ei sticla trebuia să nu fie mai groasă de 1 mm pentru ca lumina să poată să treacă prin ea. Sticla suficient de transparentă pentru a putea fi folosită ca fereastră a apărut abia în timpul Renaşterii. Sticla folosită pentru fibrele optice moderne este atât de transparentă încât, dacă oceanele ar fi fost pline cu astfel de sticlă în loc de apă, fundul oceanului s-ar vedea de la suprafaţă tot atât de clar precum se vede pământul din avion într-o zi senină.

Atenuarea luminii prin sticlă depinde de lungimea de undă a luminii (şi de alte câteva proprietăţi fizice ale sticlei). Pentru tipul de sticlă folosit la fibre optice,

atenuarea este prezentată în fig. 2-6, măsurată în decibeli pe kilometru liniar de fibră. Atenuarea în decibeli este dată de formula:

$$Atenuarea\_în\_decibeli = 10\log_{10}\frac{puterea\ transmisă}{puterea\_recepţionată}$$

De exemplu, pentru un factor de pierdere egal cu 2 rezultă o atenuare de 10 $\log_{10}2$ = 3 dB. Fig. prezintă valorile atenuării pentru lungimi de undă apropiate spectrului razelor infraroşii, care sunt folosite în practică. Lumina vizibilă are lungimi de undă puţin mai mici, de la 0.4 la 0.7 microni (1 micron este 10^{-6} metri).

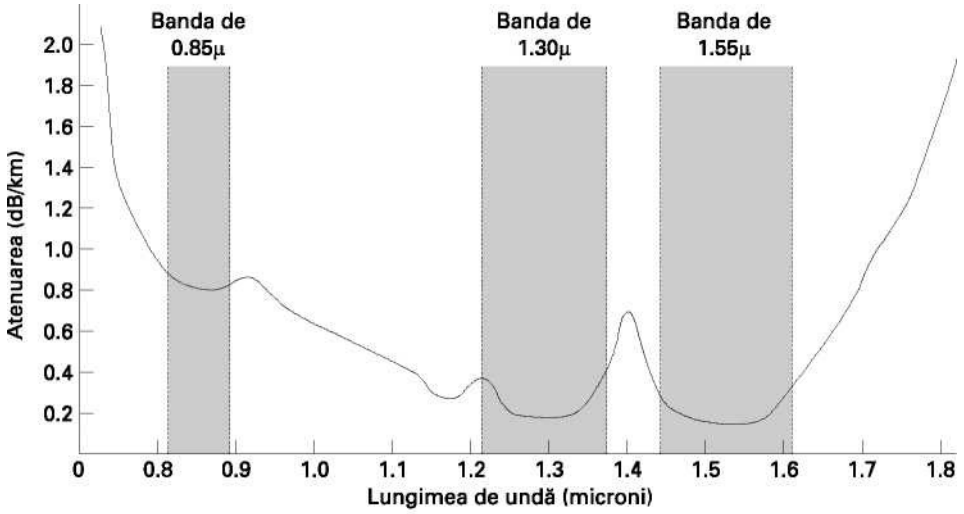


Fig. 2-6. Atenuarea luminii prin fibră în spectrul infraroșu.

Trei benzi din acest spectru sunt folosite în comunicaţii. Ele sunt centrate respectiv la 0.85, 1.3 şi 1.55 microni. Ultimele două au proprietăţi bune de atenuare (mai puţin de 5% pierderi pe kilometru). Banda de 0.85 microni are o atenuare mai mare, dar o proprietate care o avantajează este că, la această lungime de undă, laserul şi echipamentul electronic pot fi făcute din acelaşi material (arseni- ură de galiu). Toate cele trei benzi au o lărgime de bandă între 25.000 şi 30.000 GHz.

Impulsurile de lumină transmise prin fibră îşi extind lungimea în timpul propagării. Această extindere se numeşte **dispersie cromatică**, şi mărimea ei este dependentă de lungimea de undă. Un mod de a preveni suprapunerea acestor impulsuri extinse este de a mări distanţa dintre ele, dar aceasta se poate face doar prin reducerea ratei semnalului. Din fericire, s-a descoperit că, dând acestor impulsuri o formă specială, legată de reciproca cosinusului hiperbolic, se anulează toate efectele de dispersie, şi este astfel posibil să se trimită impulsuri pe mii de kilometri, fără distorsiuni semnificative ale formei. Aceste impulsuri se numesc **solitonuri**. Cercetările pentru implementarea practică a acestei soluţii de laborator sunt în plină desfăşurare.

Cablurile din fibră optică

Cablurile din fibră optică sunt similare celor coaxiale, cu singura deosebire că nu prezintă acel material conductor exterior sub forma unei plase. Fig. 2-7(a) prezintă o secţiune a unei singure fibre. În centru se află miezul de sticlă prin care se propagă lumina. În fibrele multi-mod, miezul are un diametru de 50 microni, aproximativ grosimea părului uman. În fibrele mono-mod miezul este de 8 până la 10 microni.

Miezul este îmbrăcat în sticlă cu un indice de refracţie mai mic decât miezul, pentru a păstra lumina în miez. Totul este protejat cu o învelitoare subţire din plastic. De obicei, mai multe fibre sunt grupate împreună, protejate de o teacă protectoare. Fig. 2-7(b) prezintă un astfel de cablu cu trei fibre.

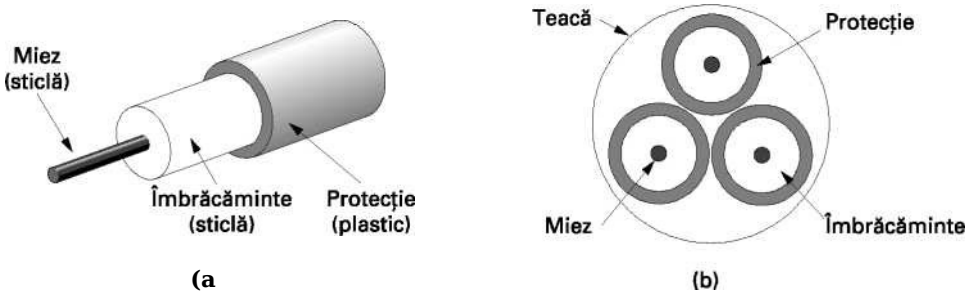


Fig. 2-7 (a) Vedere perspectivă a unei singure fibre. (b) Vedere în secţiune a unei

Fibrele terestre sunt îngropate în pământ până la adâncimi de un metru, fiind ocazional deteriorate de buldozere sau de cârtiţe. Lângă ţărm, fibrele transoceanice sunt îngropate în şanţuri cu ajutorul unui fel de plug de mare. În apele adânci, ele stau pe fundul apei, unde pot fi agăţate de traule- re de pescuit sau pot fi atacate de calmari. Fibrele pot fi conectate în trei moduri. Primul mod constă în ataşarea la capătul fibrei a unor conectori care se pot lega la un soclu pentru fibră. Conectorii pierd între 10% şi 20% din lumină, dar avantajul acestor sisteme este că sunt uşor de reconfigurat.

Al doilea mod constă în îmbinarea mecanică. Îmbinările mecanice se obţin prin ataşarea celor două capete unul lângă altul, într-un înveliş special, şi fixarea lor cu ajutorul unor clame. Alinierea se poate face prin trimitere de semnale prin joncţiune şi realizarea de mici ajustări pentru a maximiza semnalul. Unui specialist îi trebuie în jur de 5 minute să facă o îmbinare mecanică, aceasta având ca rezultat o pierdere de lumină de 10%.

A treia posibilitate este de a îmbina (topi) cele două bucăţi de fibră, pentru a forma o conexiune solidă. O îmbinare prin sudură este aproape la fel de bună ca şi folosirea unui singur fir, dar chiar şi aici, apare o mică atenuare.

Pentru toate cele trei tipuri de îmbinare poate să apară fenomenul de reflexie la punctul de îmbinare, iar energia reflectată poate interfera cu semnalul.

Criteriu	LED	Laser cu
Viteza de transfer a	Joasă	Mare
Tip de fibră	Multi-mod	Multi-mod sau uni-mod
Distanţă	Scurtă	Lungă
Durata de viaţă	Viaţă lungă	Viaţă scurtă
Sensibilitate la	Minoră	Substanţială
Cost	Cost redus	Scump

Fig. 2-8. O comparaţie între laserele semiconductoare şi LED-uri ca surse de lumină.

Pentru transmiterea semnalului se pot folosi două tipuri de surse de lumină: LED-uri (Light Emitting Diode – diodă cu emitere de lumină) şi laserul cu semiconductor. Ele au proprietăţi diferite, după cum arată fig. 2-8. Ele se pot ajusta în lungime de undă prin introducerea interferometre- lor Fabry-Perot sau Mach-Zender între sursă şi fibra optică. Interferometrele Fabry-Perot sunt simple cavităţi rezonante, formate din două oglinzi paralele. Lumina cade perpendicular pe oglinzi. Lungimea acestei cavităţi selectează acele lungimi de undă care încap în interior de un număr întreg de ori. Interferometrele Mach-Zender separă lumina în două fascicole. Cele două fascicole se propagă pe distanţe uşor diferite. Ele sunt apoi recombinate şi se află în fază doar pentru anumite lungimi de undă.

Capătul fibrei optice care recepţionează semnalul constă dintr-o fotodiodă, care declanşează un impuls electric când primeşte o rază de lumină. Timpul de răspuns tipic al unei diode este de 1ns, ceea ce limitează viteza de transfer de date la aproximativ IGbps. Pentru a putea fi detectat, un impuls luminos trebuie să aibă suficientă energie ca să evite problema zgomotului termic. Viteza de apariţie a erorilor se poate controla prin asocierea unei puteri suficient de mari a semnalului.

Reţelele din fibre optice

Fibrele optice pot fi folosite atât pentru LAN-uri cât şi pentru transmisia pe distanţe foarte lungi, deşi conectarea într-o reţea bazată pe acest mediu este mult mai complexă decât conectarea la Ethernet. O soluţie pentru a evita această problemă este prezentarea unei reţele în inel ca fiind o colecţie de legături punct la punct, aşa ca în fig. 2-9. Interfaţa fiecărui calculator lasă să treacă impulsul de lumină către următoarea legătură şi totodată are rolul unei joncţiuni în T pentru a face posibilă transmiterea şi recepţia mesajelor.

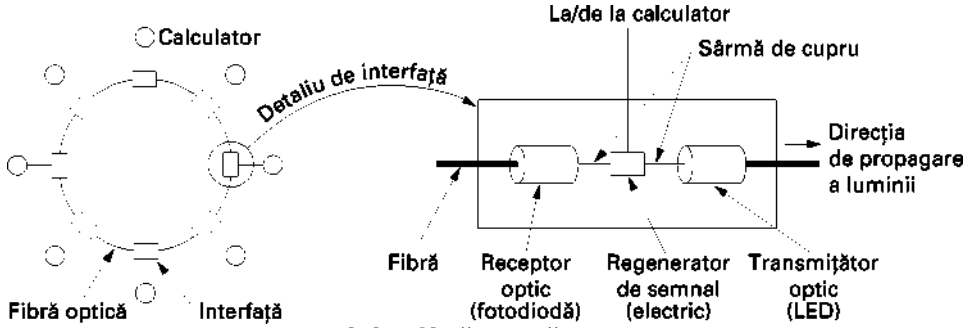


Fig. 2-9. Un inel din fibră optică cu repetoare active.

Se folosesc două tipuri de interfeţe. O interfaţă activă constă din doi conectori sudaţi pe fibra centrală. Unul din ei are la un capăt un LED sau o diodă cu laser (pentru transmisie) şi celălalt are la capăt o fotodiodă (pentru recepţie). Conectorul este complet pasiv şi este viabil, deoarece un LED sau o fotodiodă defectă nu întrerupe inelul, ci doar scoate un calculator din circuit.

Un alt model de interfaţă, prezentat în fig. 2-9, este **repetorul activ**. Lumina recepţionată este convertită într-un semnal electric, regenerat la putere maximă, dacă este atenuat şi retransmis ca semnal luminos. Interfaţa cu calculatorul este un fir de cupru obişnuit care se leagă la regeneratorul de semnal. În prezent, sunt folosite şi repetoare integral optice. Aceste echipamente nu necesită conversii de tipul optic-electric-optic, ceea ce înseamnă că pot opera la lărgimi de bandă foarte mari.

În cazul în care repetorul activ se deteriorează, inelul este întrerupt şi reţeaua nu mai funcţionează. Pe de altă parte, deoarece semnalul este regenerat de fiecare interfaţă, legăturile între două calculatoare adiacente pot avea lungimi de kilometri, practic fără nici o limitare asupra dimensiunii totale a inelului. Interfeţele pasive diminuează lumina la fiecare joncţiune, având ca efect restricţii drastice în ceea ce priveşte numărul de calculatoare ce pot fi conectate şi lungimea totală a inelului.

O topologie în inel nu este singura modalitate de a construi un LAN folosind fibre optice. Este posibilă şi o arhitectură de tip **stea pasivă**, ca aceea prezentată în fig. 2-10. În această schemă, fiecare interfaţă prezintă o fibră care face conexiunea între transmiţător şi un cilindru de siliciu, cu toate aceste fibre sudate la un capăt al cilindrului. Similar, fibrele sudate la celălalt capăt al cilindrului se conectează la fiecare receptor. Ori de câte ori o interfaţă transmite un semnal, el este difuzat în interiorul stelei pasive pentru a ilumina toţi receptorii, realizându-se astfel difuzarea. Steaua pasivă combină toate semnalele de la intrare şi transmite semnalul combinat pe toate liniile. Deoarece energia de la intrare este împărţită între toate liniile de la ieşire, numărul de noduri în reţea este limitat de senzitivitatea fotodiodelor.

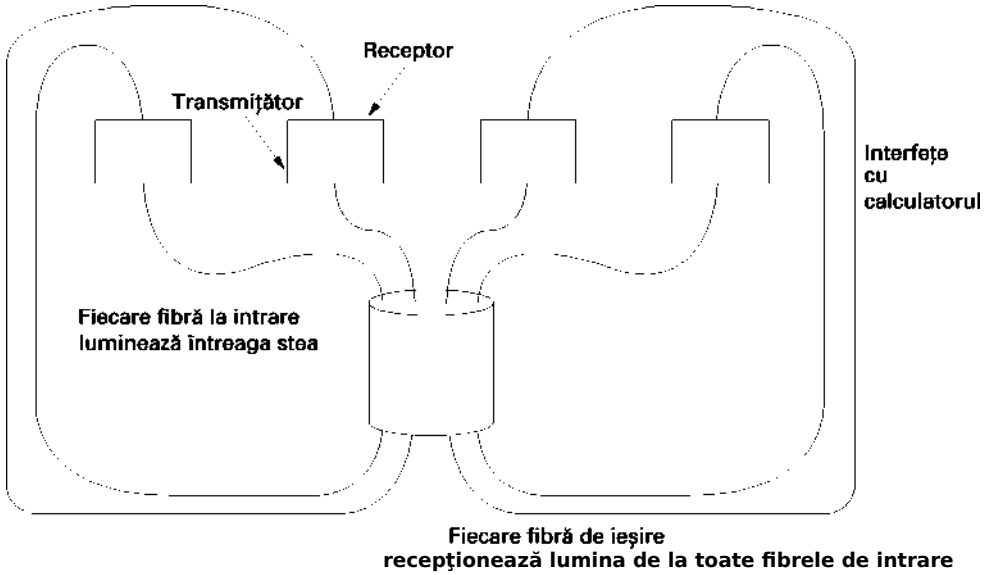


Fig. 2-10. Conectarea unei stele pasive în reţelele de fibră optică.

Comparaţie între fibrele optice şi firul de cupru

O comparaţie între fibra optică şi firele de cupru este instructivă. Fibra are multe avantaje. Mai întâi, lărgimea de bandă pe care o suportă este mai mare decât a firelor de cupru. Numai acest lucru şi ar fi suficient pentru a fi utilizată în reţelele performante. Datorită atenuării scăzute, repetoarele sunt necesare la fiecare 30 km pe liniile lungi, în comparaţie cu 5 Km pentru cupru. Fibra are avantajul că nu este afectată de şocurile electrice, de interferenţa câmpului electromagnetic sau de căderile de tensiune. De asemenea, nu este afectată nici de substanţele chimice corozive din aer, fiind ideală pentru mediile aspre din fabrici.

Destul de surprinzător, companiile de telefoane preferă fibra dintr-un alt motiv: este subţire şi foarte uşoară. Multe dintre canalele cu cabluri sunt pline până la refuz şi prin înlocuirea cuprului cu fibră se obţine ceva spaţiu, iar cuprul are o valoare excelentă pe piaţă, deoarece fabricile îl consideră un minereu de mare importanţă. De asemenea, fibra este mai uşoară decât cuprul. O mie de cabluri torsadate cu o lungime de 100 Km lungime cântăresc 8000 Kg. Două fibre au o capacitate mai mare şi cântăresc doar 100 Kg, acest lucru reducând drastic necesitatea unor echipamente mecanice scumpe care trebuie întreţinute. Pe traseele noi, fibra câştigă detaşat în faţa cuprului datorită costului de instalare foarte scăzut.

În sfârşit, fibrele nu disipă lumina şi de aceea sunt foarte dificil de interceptat. Aceste proprietăţi le oferă o excelentă securitate împotriva unor potenţiale tentative de interceptare.

Pe de altă parte, fibra este o tehnologie mai puţin familiară şi necesită o pregătire pe care nu toţi inginerii o au, iar fibrele pot fi stricate uşor dacă sunt îndoite prea mult. Deoarece transmisia optică este prin natura ei unidirecţională, comunicaţiile bidirecţionale necesită fie două fibre, fie două benzi de frecvenţă diferite pe aceeaşi fibră. În sfârşit, interfeţele pentru fibră costă mai mult decât interfeţele electrice. Nu mai puţin adevărat este faptul că toate comunicaţiile de date pe lungimi mai mari de câţiva metri se vor face în viitor cu fibre. Pentru o

discuţie asupra tuturor aspectelor fibrelor optice şi asupra reţelelor construite cu ele, vedeţi (Hecht, 2001).

2.3 COMUNICAŢIILE FĂRĂ FIR

Epoca noastră a generat dependenţa de informaţie: oameni care au nevoie să fie în permanenţa conectaţi la informaţii. Pentru aceşti utilizatori mobili, cablul torsadat, cablul coaxial şi fibrele optice nu sunt de nici un folos. Ei au nevoie de date pentru calculatoarele lor portabile, fără a fi legaţi de infrastructura comunicaţiilor terestre. Pentru aceşti utilizatori comunicaţiile fără fir reprezintă soluţia optimă. În secţiunile ce urmează, vom discuta la modul general asupra comunicaţiilor fără fir, deoarece acestea au multe alte aplicaţii importante în afara serviciilor de conectare oferite utilizatorilor care doresc să navigheze pe WEB de pe plajă.

Sunt voci care susţin că viitorul rezervă numai două tipuri de comunicaţii: prin fibre optice şi fără fir. Toate calculatoarele, faxurile, telefoanele fixe (nemobile) vor folosi fibre, iar cele mobile vor folosi comunicaţia fără fir.

Comunicaţiile fără fir sunt avantajoase chiar şi pentru echipamentele fixe, în anumite împrejurări. De exemplu, în cazul în care conectarea unei clădiri cu ajutorul fibrei este dificilă datorită terenului (munţi, jungle, mlaştini etc.), comunicaţia fără fir poate fi mai bună. Este de remarcat faptul că

sistemele moderne de comunicaţie digitală fără fir au apărut în Insulele Hawaii, unde utilizatorii erau despărţiţi de mari întinderi de apă din oceanul Pacific, sistemul telefonic fiind inadecvat.

2.3.1 Spectrul electromagnetic

Atunci când electronii se află în mişcare, ei creează unde electromagnetice care se pot propaga prin spaţiu (chiar şi în vid). Aceste unde au fost prezise de fizicianul britanic James Clerk Maxwell în 1865 şi au fost observate pentru prima dată de fizicianul german Heinrich Hertz în 1887. Numărul de oscilaţii ale unei unde într-o secundă poartă numele de **frecvenţă**, *f,* şi este măsurată în **Hz** (în onoarea lui Heinrich Hertz). Distanţa dintre două maxime (sau minime) consecutive este numită **lungime de undă.** Notaţia universală a lungimii de undă este 7 (lambda).

Când o antenă dimensionată corespunzător este ataşată unui circuit, undele electromagnetice pot fi difuzate eficient şi interceptate de un receptor, aflat la o anumită distanţă. Acest principiu stă la baza tuturor comunicaţiilor fără fir.

În vid, toate undele electromagnetice se transmit cu aceeaşi viteză, indiferent de frecvenţă, Această viteză, numită de obicei **viteza luminii**, *c,* este de aproximativ de 3×10^8 m/s, sau aproape 1 picior (30 cm) pe nanosecundă. (Ar fi o ideea redefinirea *piciorului (*eng: *foot)* ca fiind distanţa pe care o parcurge lumina în vid într-o nanosecunda, mai degrabă decât definirea pe baza mărimii pantofului unui rege oarecare mort demult). În cupru sau în fibră, viteza scade la aproape 2/3 din această valoare şi devine uşor dependentă de frecvenţa undei. Viteza luminii este viteza

maximă care poate fi atinsă – nici un obiect sau semnal nu se deplasează vreodată cu o viteza mai mare ca aceasta.

Relaţia fundamentală dintre *f*, *7* şi *c* (în vid) este

$$2\,f = c \qquad (2\text{-}2)$$

Deoarece *c* este o constantă, ştiind *f* putem afla 7. dar şi invers. Ca o regulă clară, reţineţi că atunci când *2* este în metri şi*f* este în MHz, *2 f*= 300. De exemplu, undele cu frecvenţa de 100 MHz au lungimea de undă de aproape 3 metri, cele cu frecvenţa de 1000 MHz au lungimea de undă de 0.3 metri, iar cele cu lungimea de undă de 0.1 metri au frecvenţa de 3000 MHz.

În fig. 2-11 este prezentat spectrul electromagnetic. Domeniile corespunzătoare undelor radio, microundelor, undelor infraroşii şi luminii vizibile din spectru pot fi folosite pentru transmiterea informaţiei prin modularea amplitudinii, frecvenţei sau fazei undelor. Lumina ultravioletă, razele X şi razele gama ar fi chiar mai performante datorită frecvenţelor lor mai înalte, dar ele sunt greu de produs şi de modulat, nu se propagă bine prin clădiri şi sunt periculoase pentru fiinţele vii. Benzile listate în partea de jos a fig. 2-11 sunt numele oficiale ITU şi se bazează pe lungimile de undă, LF acoperind intervalul de la 1 Km la 10 Km (aproximativ de la 30 KHz la 300 KHz). Termenii de LF, MF şi HF se referă la frecvenţele joase, medii şi înalte, respectiv. Este evident că atunci când au fost date aceste nume, nimeni nu se aştepta ca frecvenţe mai mari de 10 MHz să se folosească vreodată. Benzile mai înalte au fost numite mai târziu benzi de frecvenţă Foarte, Ultra, Super, Extrem şi Extraordinar de înalte. Dincolo de aceste frecvenţe nu mai există denumiri consacrate, dar am putea să folosim expresii de genul frecvenţe Incredibil, Uimitor sau Miraculos de înalte.

Cantitatea de informaţie pe care o undă electromagnetică o poate transporta este legată de lărgimea ei de bandă. Folosind tehnologia curentă, este posibil să codificăm câţiva biţi pe Hertz la frecvenţe joase şi deseori până la 8 biţi pe Hertz la frecvenţe înalte; în concluzie, un cablu torsadat cu lărgimea de bandă de 750MHz poate transporta date de ordinul a câţiva gigabiţi/s. Din fig. 2-11 ar trebui să reiasă de acum foarte clar de ce profesioniştii din domeniul reţelelor apreciază atât de mult fibrele optice.

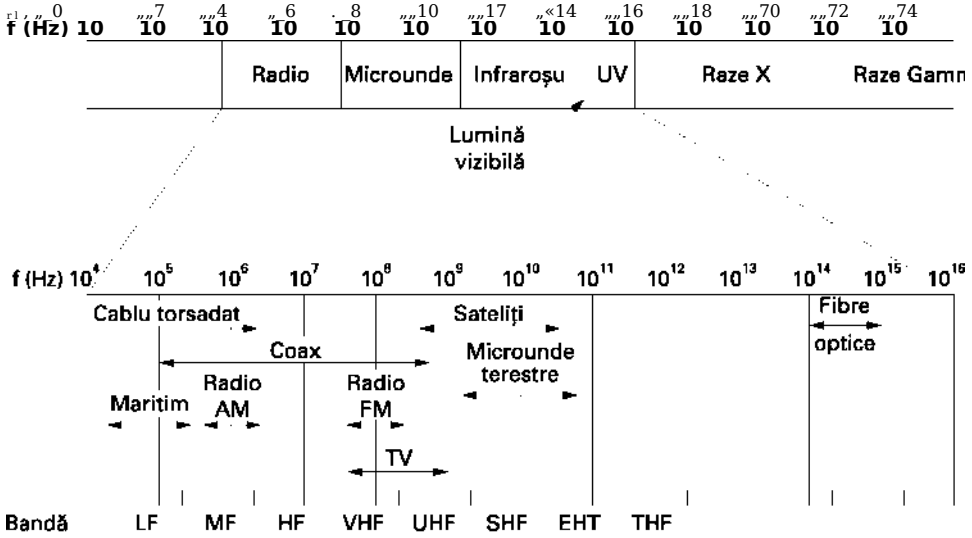


Fig. 2-11. Spectrul electromagnetic aşa cum este folosit în comunicaţii.

Dacă rezolvăm Ec. (2-2) pentru *f* şi o diferenţiem în raport cu lungimea de undă, obţinem

$$\frac{df}{d\lambda} = -\frac{c}{\lambda^2}$$

Dacă trecem la diferenţe finite în loc de diferenţiale şi alegem doar valorile pozitive, obţinem:

$$\Delta f = \frac{c\,\Delta\lambda}{\lambda^2} \qquad (2-$$

Astfel, fiind dată lărgimea unei benzi de lungimi de undă, AX, putem calcula banda de frecvenţă corespunzătoare, *Af* şi, în continuare, viteza de transfer de date pe care banda o poate produce. Cu cât banda este mai largă, cu atât creşte viteza de transfer a datelor. De exemplu, să considerăm banda de 1.30 microni din fig. 2-6. Aici avem X = 1.3 x 10^{-6} şi AX = 0.17 x 10^{-6}, cu A*f* aproape 30 THz. La , să zicem, 8 biţi/Hz, avem 240 Tbps.

Majoritatea transmisiilor folosesc o bandă îngustă de frecvenţă (A*f* / *f* << 1) pentru a obţine cea mai bună recepţie (cât mai mulţi Watts/Hz). Totuşi, banda largă este folosită în două situaţii. În cazul utilizării metodei **salturilor de frecvenţă într-un spectru larg** (**frequency hopping spread spectrum**), transmiţătorul sare de la o frecvenţă la alta de sute de ori pe secundă. Ea este foarte populară în comunicaţiile armatei, deoarece face transmisia greu de detectat şi aproape imposibil de bruiat. De asemenea, oferă un grad ridicat de imunitate la atenuarea multi-căi, deoarece semnalul direct ajunge întotdeauna primul la receptor. Semnalele reflectate urmăresc o cale mai lungă şi ajung mai târziu. Până atunci receptorul a schimbat deja frecvenţa şi nu mai primeşte semnale cu frecvenţa anterioară, eliminând astfel orice interferenţă între semnalele directe şi cele reflectate. În ultimii ani, această tehnică a fost aplicată comercial – de exemplu, pentru 802.11 şi pentru Bluetooth.

Ca o curiozitate, tehnica a fost co-inventată de Hedy Lamarr, o senzuală actriţă

născută în Austria, prima femeie care a apărut goală intr-un film (în 1933, în filmul cehesc Extase). Primul ei soţ era fabricant de armament, şi i-a spus cât de uşor era să blochezi undele radio, folosite pe atunci pentru ghidarea torpilelor. Îngrozită când a descoperit că el îi vindea arme lui Hitler, ea s-a deghizat în servitoare pentru a fugi şi a zburat la Hollywood pentru a-şi continua cariera de actriţă de film. În timpul liber, a inventat metoda salturilor de frecvenţe pentru a-i ajuta pe aliaţi în război. Schema ei folosea 88 de frecvenţe, numărul de clape (şi frecvenţe) de la pian. Pentru invenţia lor, ea şi prietenul ei, compozitorul de musical-uri George Antheil, au primit patentul U.S. cu numărul 2.292.387 . Totuşi, nu au reuşit sa convingă marina americană că invenţia lor putea fi pusă în practică şi nu au primit niciodată nici un fel de onoruri. Abia peste ani, după ce patentul expirase, metoda a devenit o tehnică populară.

Şi cealaltă formă de spectru larg, **spectru larg cu succesiune directă** (**direct sequence spread spectrum**), metodă care împrăştie semnalul pe o bandă de frecvenţă largă câştigă popularitate în lumea comercială. În particular, unele telefoane mobile din cea de-a doua generaţie folosesc această tehnică, şi va deveni predominantă pentru generaţia a treia, datorită eficienţei spectrale bune, imunităţii la zgomot şi a altor proprietăţi. Este folosită şi în unele LAN-uri fără fir. Vom reveni la discuţia despre spectrul larg mai târziu în acest capitol. Pentru o istorie fascinantă şi detaliată a comunicaţiilor în spectru larg, vezi (Scholtz, 1982).

Pentru moment, vom considera că toate transmisiunile folosesc o bandă de frecvenţă îngustă. Vom discuta despre modul în care diverse părţi ale spectrului electromagnetic din fig. 2-11 sunt folosite, începând cu banda radio.

2.3.2 Transmisia radio

Undele radio sunt uşor de generat, pot parcurge distanţe mari, penetrează cu uşurinţă zidurile clădirilor, fiind larg răspândite în comunicaţii, atât interioare cât şi exterioare. De asemenea, undele radio sunt omnidirecţionale, ceea ce înseamnă că se pot propaga în orice direcţie de la sursă, deci nu este nevoie de o aliniere fizică a transmiţătorului cu receptorul.

Uneori această proprietate de propagare omnidirecţională este bună, alteori nu. În anii ‘70, General Motors a decis să echipeze noile sale Cadillac-uri cu un calculator care să prevină blocarea frânelor. Atunci când şoferul apăsa pedala de frână, calculatorul frâna treptat, în loc să preseze frâna complet. Într-o zi frumoasă de vară, un ofiţer de pe o autostradă din Ohio a început să-şi folosească staţia radio mobilă pentru a chema sediul central şi deodată un Cadillac situat în apropriere a început să se cabreze ca un cal sălbatic. Atunci când ofiţerul a oprit maşina, şoferul a pretins că el nu a făcut nimic, dar maşina a înnebunit.

În cele din urmă a reieşit că lucrurile se petreceau după un anumit tipar: maşinile Cadillac erau scăpate uneori de sub control, dar numai pe marile autostrăzi din Ohio şi numai când o patrulă de poliţie era în zonă. Pentru o foarte lungă perioadă de timp, cei de la General Motors nu au înţeles de ce maşinile mergeau foarte bine în toate celelalte state, ca şi pe străzile secundare din Ohio. După îndelungi căutări au descoperit că, în Cadillac, cablajul forma o antenă foarte bună pentru frecvenţa folosită de noul sistem radio al poliţiei rutiere din Ohio.

Proprietăţile undelor radio sunt dependente de frecvenţă. La frecvenţe joase, undele radio se propagă bine prin obstacole, dar puterea semnalului scade mult odată cu distanţa de la sursă, aproximativ cu $1/r^2$ în aer. La frecvenţe înalte, undele radio tind să se propage în linie dreaptă şi să ricoşeze din obstacole. De asemenea, sunt absorbite de ploaie. Mai mult, toate frecvenţele radio sunt supuse la interferenţe datorate motoarelor şi altor echipamente electrice.

Datorită capacităţii undelor radio de a se propaga pe distanţe mari, interferenţa dintre utilizatori devine o problemă. Acesta este principalul motiv pentru care toate guvernele acordă cu foarte mare atenţie licenţele pentru utilizatorii de transmiţătoare radio, cu o singură excepţie (discutată mai jos).

În benzile de frecvenţă foarte joasă (VLF), joasă (LF) şi medie (MF), undele radio se propagă la sol, după cum este ilustrat în fig. 2-12(a). Aceste unde pot fi detectate pe distanţe de până la aproximativ 1000 Km pentru frecvenţele mai joase şi pe distanţe mai mici pentru cele mai înalte. Pentru difuzarea undelor radio AM se foloseşte banda MF, acesta fiind motivul pentru care staţia radio AM din Boston nu poate fi auzită cu uşurinţă în New York. Undele radio în această bandă trec uşor prin clădiri, fiind astfel posibilă utilizarea radiourilor portabile în spaţii interioare. Problema principală care apare la comunicaţia de date la aceste frecvenţe este lărgimea mică a benzii pe care o oferă [vezi ecuaţia (2-2)].

În benzile de frecvenţă înaltă şi foarte înaltă, undele de la sol tind să fie absorbite de pământ. Totuşi, undele care ating ionosfera (un strat de particule care înveleşte atmosfera la o înălţime de 100 până la 500 Km), sunt refractate de aceasta şi trimise înapoi spre pământ, după cum arată fig. 2-12(b). În anumite condiţii atmosferice, semnalele pot parcurge acest drum de mai multe ori.

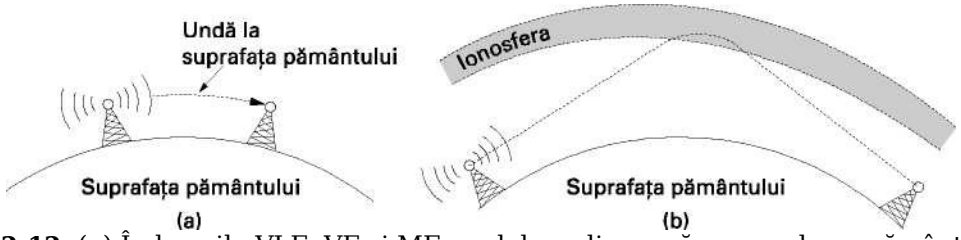


Fig. 2-12. (a) În benzile VLF, VF şi MF, undele radio urmăresc curbura pământului. (b) În banda HF undele revin din ionosferă.

Operatorii radio amatori folosesc aceste benzi pentru a realiza convorbiri la mare distanţă. De asemenea, comunicaţiile armatei se desfăşoară în benzile de frecvenţe înalte şi foarte înalte.

2.3.3 Transmisia prin microunde

Peste 100 MHz, undele se propagă în linii aproximativ drepte şi pot fi, din acest motiv, direcţio- nate. Concentrând toată energia într-un fascicol îngust, cu ajutorul unei antene parabolice (ca o antenă de satelit obişnuită) rezultă o valoare mult mai ridicată a raportului semnal-zgomot, dar antenele care transmit şi cele care recepţionează trebuie să fie aliniate cu precizie una cu alta. În plus, faptul că aceste

unde sunt orientate permite ca mai multe transmiţătoare să fie aliniate şi să comunice cu mai multe receptoare aflate în linie, fără interferenţe, cu condiţia să se respecte câteva reguli de distanţare. Înaintea fibrelor optice, microundele au format, timp de decenii, inima sistemului telefonic de comunicaţie pe distanţe mari. De fapt, MCI, una dintre primele competitoare AT&T , şi-a construit întregul sistem prin comunicaţii cu microunde, din turn în turn, la distanţe de zeci de kilometri între ele. Chiar şi numele companiei reflectă acest lucru (MCI înseamnă Microwave Communications, Inc.). MCI a trecut apoi pe fibră şi a fuzionat cu WorldCom.

Datorită faptului că microundele se propagă în linii drepte, dacă turnurile sunt foarte depărtate, atunci în calea lor stă chiar Pământul (gândiţi-vă la o legătură între San Francisco şi Amsterdam). În consecinţă trebuie instalate repetoare din loc în loc. Cu cât turnurile sunt mai înalte, cu atât distanţa dintre repetoare este mai mare. Distanţa dintre repetoare creşte direct proporţional cu radicalul înălţimii turnului. Pentru turnuri cu o înălţime de 100 m, distanţa dintre repetoare poate fi de 80 km.

Spre deosebire de undele radio de frecvenţe joase, microundele nu trec cu uşurinţă prin zidurile clădirilor. În plus, cu toate că unda poate fi bine direcţionată la transmiţător, apare o divergenţă în spaţiu. Unele unde pot fi refractate de straturile atmosferice joase şi pot întârzia mai mult decât undele directe. Undele întârziate pot sosi defazate faţă de unda directă, anulând astfel semnalul. Acest efect este numit **atenuare multi-căi** (**multipath fading**) şi constituie deseori o problemă serioasă. Este dependentă de vreme şi de frecvenţă. Unii operatori păstrează nefolosit un procent de 10 la sută din canalul propriu pentru a putea comuta pe acesta atunci când atenuarea multi-căi le anulează temporar anumite benzi de frecvenţă.

Cererea de benzi este din ce în ce mai mare şi duce operatorii către frecvenţe tot mai înalte. Benzi de până la 10 GHz sunt acum uzuale, dar la aproape 8 GHz apare o nouă problemă: absorbţia de către apă. Aceste unde au lungimi de doar câţiva centimetri şi sunt absorbite de ploaie. Acest efect ar fi potrivit pentru cineva care ar fi încercat să construiască un imens cuptor cu microunde în aer liber, dar pentru comunicaţii este o problemă dificilă. La fel ca şi în cazul atenuării multi-căi, singura soluţie posibilă este întreruperea legăturilor acolo unde plouă şi găsirea unei rute alternative.

Comunicaţiile cu microunde sunt atât de larg folosite de telefonia pe distanţe mari, telefoanele celulare, televiziune şi altele, încât a apărut o criză în ceea ce priveşte spectrul. Microundele au mai multe avantaje semnificative faţă de fibră. Cel mai important avantaj este că nu sunt necesare drepturi de acces la drum, cumpărând un mic teren la fiecare 50 Km şi montând un turn pe el, se poate ocoli sistemul telefonic şi se poate realiza o comunicare directă. Astfel a reuşit MCI să pornească atât de rapid ca o companie de telefoane pe distanţe mari. (Sprint a aplicat o altă tactică : a fost formată de Southern Pacific Railroad (căile feroviare sudice), care deja deţinea destule drepturi de acces şi tot ce a avut de făcut a fost să îngroape fibra lângă şine).

Comunicaţiile cu microunde, prin comparaţie cu alte medii de transmisie, sunt ieftine. Preţul ridicării a două turnuri simple (doi stâlpi înalţi asiguraţi cu patru cabluri) şi de montare a unei antene pe fiecare turn, poate fi mai mic decât preţul îngropării a 50 de Km de fibră într-o zonă urbană foarte populată sau peste un munte şi poate fi mai mic decât costul închirierii fibrei de la o companie telefonică,

mai ales atunci când acestea nu au plătit încă integral cuprul care a fost înlocuit cu fibră.

Politica din domeniul spectrului electromagnetic

Pentru a preveni haosul general, există convenţii naţionale şi internaţionale care reglementează modul de folosire al frecvenţelor – cine ce frecvenţă foloseşte. Deoarece toată lumea doreşte o rată de transfer cât mai mare, toată lumea doreşte cât mai mult din spectru. Guvernele naţionale alocă spectru pentru radio AM şi FM, televiziune, telefoane mobile, ca de altfel şi pentru companii telefonice, poliţie, marină, navigatori, armată, guvern şi mulţi alţi clienţi competitori. La nivel internaţional agenţia ITU-R (WARC) încearcă să coordoneze această alocare, astfel încât să poată fi construite dispozitive care să funcţioneze în diverse ţări. Totuşi, ţările nu sunt obligate să respecte recomandările ITU, iar FCC (Comisia Federală de Comunicaţii), comisia care se ocupă cu alocarea de spectru în Statele Unite, a respins de câteva ori recomandările ITU (de obicei pentru că acestea cereau unui grup puternic din punct de vedere politic să renunţe la o parte din spectru).

Chiar şi atunci când o parte din spectru a fost alocată unei anumite utilizări, cum ar fi telefonia mobilă, apare discuţia legată de modul de alocare a frecvenţelor către companiile de telecomunicaţii. În trecut, erau utilizaţi trei algoritmi. Cel mai vechi algoritm, adesea denumit **concursul de frumuseţe**, cerea fiecărei companii de telecomunicaţii să explice de ce propunerea sa serveşte cel mai

bine interesul public. Oficialii guvernului decideau apoi care dintre poveşti le place cel mai mult. Posibilitatea ca unul dintre oficialii să premieze compania favorită cu o proprietate valorând miliarde de dolari ducea adesea la mită, corupţie, nepotism şi chiar mai rău. Mai mult, chiar şi un oficial cinstit al guvernului, căruia i se părea că o companie străină ar face o treabă mai bună decât companiile naţionale, ar fi avut de dat multe explicaţii.

Această observaţie a condus la algoritmul 2: organizarea unei loterii între companiile interesate. În acest caz, problema este că pot participa la loterie chiar şi companii care nu au nici o intenţie de utilizare a spectrului. Să zicem că un restaurant cu servire rapidă sau un lanţ de magazine de pantofi ar câştiga: acesta ar putea revinde spectrul unei companii de telecomunicaţii, cu un profit imens şi fără riscuri.

Scandaluri furtunoase datorate unor astfel de situaţii alarmante, deşi aleatoare, au dus la critici severe din partea multora, ceea ce a condus la algoritmul 3: licitarea benzii de frecvenţă şi atribuirea ei celui care dă mai mult. În anul 2000, când Anglia a licitat frecvenţele necesare pentru generaţia a treia de sisteme mobile, se aşteptau să primească în jur de 4 miliarde de dolari. De fapt, au primit 40 de miliarde de dolari, deoarece companiile de telecomunicaţii au intrat într-o frenezie molipsitoare, fiind speriate de moarte că vor rata trenul telefoniei mobile. Acest eveniment a declanşat lăcomia guvernelor din jur şi le-a inspirat să organizeze propriile licitaţii. A funcţionat, dar a şi lăsat unele dintre companiile de telecomunicaţii cu foarte multe datorii, aproape de faliment. Chiar şi în cele mai bun variante, vor trece mulţi ani până când acestea vor reuşi să acopere taxele de licenţă.

O modalitate cu totul diferită de a aborda problema alocării frecvenţelor este să nu fie alocate în nici un fel. Toată lumea este lăsată să transmită cât doreşte, fiind reglementată doar puterea folosită, astfel încât staţiile să aibă o rază de acţiune suficient de scurtă ca să nu interfereze între ele. În consecinţă, cele mai multe guverne au pus deoparte câteva benzi de frecvenţă, numite benzi **ISM (Industriale, Ştiinţifice, Medicale)** pentru utilizare nelicenţiată. Sistemele de deschidere de uşi de garaj, telefoane fără fir, jucării telecomandate, mouse-uri fără fir, şi multe alte aparate casnice fără fir folosesc benzile ISM. Pentru a minimiza interferenţa între aceste dispozitive necoordonate, FCC a impus ca toate dispozitivele din benzile ISM să folosească tehnici de spectru larg. Reguli asemănătoare sunt aplicate şi în alte ţări.

Fig. 2-13. Benzile ISM în Statele Unite.

Localizarea benzilor ISM diferă oarecum de la o ţară la alta. În Statele Unite, de exemplu, dispozitivele a căror putere este sub 1 Watt pot folosi benzile din fig. 2.13 fără a cere licenţă FCC. Banda 900Mhz este cea mai bună, dar este aglomerată şi nu este disponibilă în întreaga lume. Banda 2.4GHz este disponibilă în cele mai multe ţări, dar este supusă interferenţei cu instalaţiile radar sau cuptoarele cu microunde. Bluetooth şi unele dintre LAN-urile fără fir conforme 802.11 operează pe această bandă. Banda 5.7GHz este nouă şi relativ nedezvoltată, astfel că echipamentele pentru ea sunt scumpe; dar pentru că 802.11 o foloseşte, va deveni rapid mai populară.

2.3.4 Undele infraroşii şi milimetrice

Undele infraroşii şi milimetrice neghidate sunt larg folosite pentru comunicaţiile pe distanţe mici. Telecomenzile pentru televizoare, aparate video sau combine muzicale folosesc comunicaţiile în infraroşu. Ele sunt relativ direcţionale, ieftine şi uşor de construit, dar au un dezavantaj major: nu penetrează obiectele solide (încercaţi să staţi între telecomandă şi televizor şi vedeţi dacă telecomanda mai dă comenzi televizorului). În general, pe măsură ce ne deplasăm de la undele radio lungi către lumina vizibilă, undele se comportă din ce în ce mai mult ca lumină şi din ce în ce mai puţin ca unde radio.

Pe de altă parte, faptul că razele infraroşii nu trec prin obiecte constituie un avantaj. Aceasta înseamnă că un sistem cu infraroşii dintr-o cameră a unei clădiri nu va interfera cu un sistem similar situat în camerele sau clădirile adiacente: nu poţi controla televizorul vecinului tău cu telecomanda ta. Mai mult, sistemele de protecţie cu radiaţii infraroşii asigură o mai bună protecţie împotriva interceptărilor în raport cu sistemele radiofonice, tocmai din acest motiv. Datorită acestor motive, pentru operarea unui sistem cu infraroşii nu este necesară procurarea unei licenţe, spre deosebire de sistemele radiofonice, care trebuie să deţină o licenţă.

2.3.5 Transmisia undelor luminoase

Semnalele optice neghidate au fost folosite secole întregi. Înaintea faimoasei lui călătorii, Paul Revere a folosit semnalizarea optică binară de la Old North Church. O aplicaţie mai modernă este conectarea reţelei locale între două clădiri prin intermediul laserelor montate pe acoperişurile acestora. Semnalizarea optică folosind laserul este inerent unidirecţională, deci fiecare clădire are nevoie de propriul ei laser şi de propria ei fotodiodă. Această schemă oferă o bandă foarte largă la un cost foarte redus. De asemenea, este uşor de instalat şi, spre deosebire de microunde, nu necesită o licenţă FCC. Puterea laserului, un fascicol foarte îngust, este aici o slăbiciune. Îndreptarea unui fascicol de lumină de 1mm lăţime către o ţintă de lăţimea unui ac cu gămălie aflată la 500 de metri depărtare necesită o tehnică de vârf. De obicei sunt folosite lentile pentru o uşoară defocalizare a fascicolului.

Un alt dezavantaj este că fascicolul laser nu penetrează ploaia şi ceaţa groasă, dar în mod normal ele funcţionează bine în zilele însorite. Totuşi, autorul a participat odată într-un hotel modern din Europa la o conferinţă la care organizatorii conferinţei s-au gândit să pună la dispoziţie o cameră plină cu terminale, în care

participanţii să-şi poată citi poşta electronică în timpul prezentărilor plictisitoare. Deoarece PTT-ul local nu dorea să instaleze un număr mare de linii telefonice doar pentru 3 zile, organizatorii au montat pe acoperiş un laser orientat către clădirea departamentului de calculatoare al universităţii de calculatoare aflată la o distanţa de câţiva kilometri. Ei l-au testat cu o noapte înainte şi totul a decurs perfect. În dimineaţa următoare, într-o zi însorită, la ora 9, legătura a căzut şi a rămas nefuncţională toata ziua. Seara, organizatorii au testat-o din nou cu atenţie şi a funcţionat încă o dată perfect. Acelaşi lucru s-a întâmplat încă două zile la rând.

După conferinţă, organizatorii au descoperit problema. Căldura datorată soarelui din timpul zilei a determinat naşterea unor curenţi de convecţie din acoperişul clădirii, ca în fig. 2-14. Acest aer turbulent a deviat fascicolul şi l-a făcut să oscileze în jurul detectorului. Această „vedere” atmosferică face ca stelele să pâlpâie (acesta este motivul pentru care astronomii îşi pun telescoapele pe vârful munţilor – ca să fie cât se poate de mult deasupra atmosferei). Efectul respectiv este responsabil şi pentru „tremurul” şoselei într-o zi însorită şi a imaginii vălurite deasupra unui radiator fierbinte.

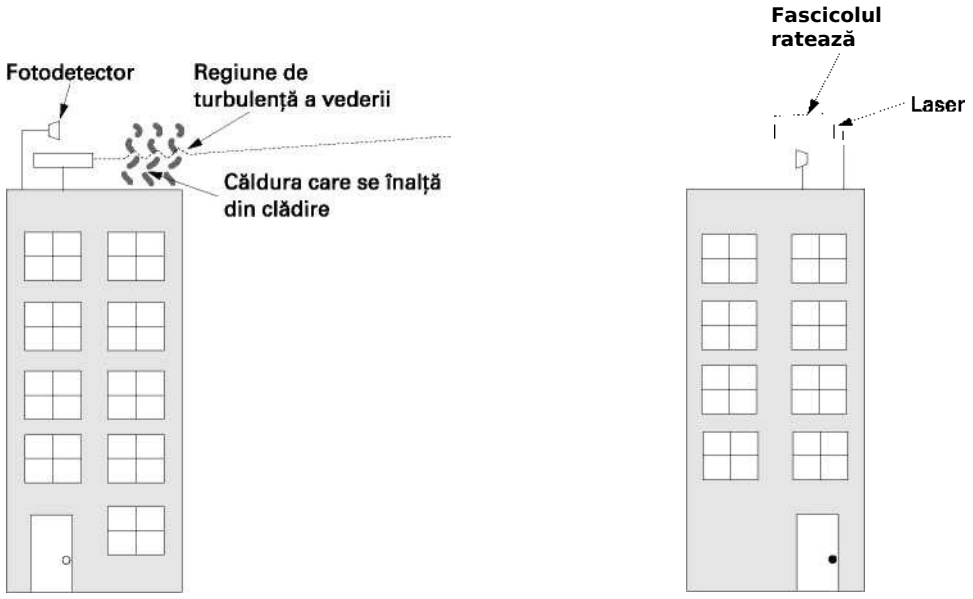


Fig. 2-14. Curenţii de convecţie pot interfera cu sistemele de comunicaţie prin laser. Aici este prezentat un sistem bidirecţional, cu două lasere.

2.4 SATELIŢI DE COMUNICAŢIE

În anii 1950 şi la începutul anilor 1960, oamenii au încercat să construiască sisteme de comunicaţie pe baza reflectării semnalelor de către baloanele meteorologice metalizate. Din nefericire, semnalele recepţionate erau prea slabe ca să poată fi folosite practic la ceva. Apoi, Marina S.U.A. a observat pe cer un fel de balon meteorologic permanent – Luna – şi a construit, pe baza reflectării semnalelor de către Lună, un sistem operaţional pentru comunicaţii navă-ţărm.

Progresul în domeniul comunicaţiilor celeste a trebuit să mai aştepte până când a fost lansat primul satelit de comunicaţii, în 1962. Principala diferenţă între un satelit artificial şi unul natural este aceea că satelitul artificial poate amplifica semnalele înainte de a le transmite înapoi, transformând ceea ce era doar o curiozitate într-un sistem puternic de comunicaţie.

Sateliţii de comunicaţie au câteva proprietăţi interesante, care îi fac tentanţi pentru multe aplicaţii. Un satelit de comunicaţie poate fi gândit ca un mare repetor de microunde, aflat în cer. Acesta conţine mai multe **dispozitive de recepţie-transmisie automată** (**transporder**); fiecare dintre ele ascultă pe o anume porţiune din spectru, amplifică semnalul recepţionat şi apoi îl redifuzează pe o altă frecvenţă, pentru a evita interferenţa cu semnalul recepţionat. Unda descendentă poate fi

difuzată, acoperind astfel o fracţiune substanţială din suprafaţa Pământului sau poate fi concentrată, caz în

care va acoperi numai o zonă de câteva sute de kilometri în diametru. Acest mod de operare este cunoscut şi sub numele de **ţeavă îndoită** (**bent pipe**).

Conform legii lui Kepler, perioada de rotaţie a unui satelit variază cu puterea 3/2 a razei orbitei. Cu cât un satelit este mai sus, cu atât este mai lungă perioada. În apropierea suprafeţei Pământului, perioada este de aproximativ 90 de minute. În consecinţă, sateliţii cu orbita mai mică dispar din raza vizuală destul de repede, astfel că mulţi dintre ei sunt necesari pentru a oferi acoperire continuă. La o altitudine de aproximativ 35.800 km, perioada unui satelit este de 24 de ore. La o altitudine de aproximativ 384.000 km, perioada unui satelit este de aproape o lună, aşa după cum poate confirma oricine care a urmărit luna în mod regulat.

Perioada unui satelit este importantă, dar nu este singurul criteriu în determinarea locului în care va fi plasat satelitul. Un alt criteriu este prezenţa centurilor lui Van Allen, straturi de particule foarte încărcate prinse în câmpul magnetic al pământului. Aceşti factori conduc la formarea a trei regiuni în care sateliţii pot fi plasaţi în siguranţă. Aceste regiuni şi unele dintre proprietăţile lor sunt ilustrate în fig. 2-15. În continuare vom descrie pe scurt sateliţii care sunt plasaţi în fiecare dintre aceste regiuni.

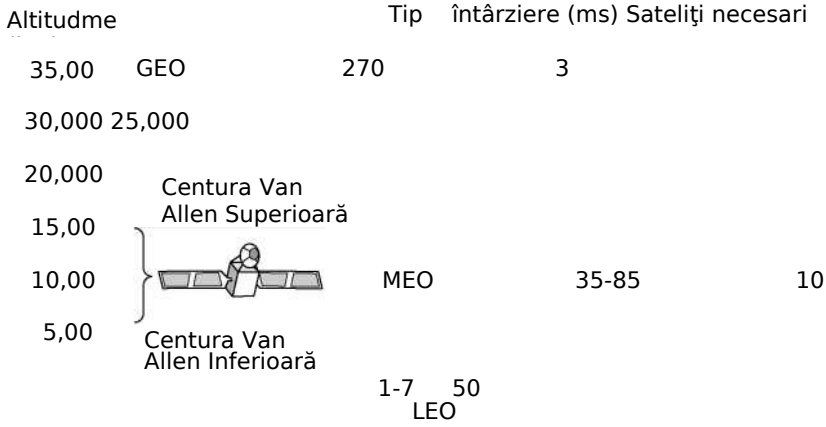


Fig. 2-15. Sateliţii de comunicaţie şi câteva dintre proprietăţile lor, inclusiv altitudinea, întârzierea pentru un drum dus-întors şi numărul de sateliţi necesari pentru acoperirea globală.

2.4.1 Sateliţi geostaţionari

În 1945, scriitorul de proză ştiinţifico-fantastică Arthur C. Clarke a calculat că un satelit aflat la altitudinea de 35.800 km pe o orbită ecuatorială circulară apare ca staţionar pe cer, astfel încât nu este nevoie să fie urmărit (Clarke, 1945). El a mers mai departe şi a descris un sistem complet de comunicare care folosea aceşti **sateliţi geostaţionari**, inclusiv orbitele lor, panourile solare, frecvenţele radio şi procedurile de lansare.

Din păcate, a ajuns la concluzia că sateliţii nu erau o soluţie practică datorită imposibilităţii de a lansa pe orbită amplificatoare cu tub catodic, acestea fiind fragile şi mari consumatoare de energie, aşa că nu a mai continuat pe aceasta idee, deşi a

scris câteva povestiri ştiinţifico-fantastice bazate pe ea.

Inventarea tranzistorului a schimbat toate acestea şi a fost lansat primul satelit artificial pentru comunicaţii, Telstar, în iulie 1962. Din acel moment, sateliţii pentru comunicaţii au devenit o afacere de miliarde de dolari şi totodată singurul aspect al explorării spaţiului cosmic care a devenit foarte profitabil. Aceşti sateliţi care zboară la altitudini mari sunt adeseori numiţi sateliţi **GEO** (**Geostationary Earth Orbit**, rom: cu orbită geostaţionară terestră).

Pentru a evita interferenţa, în condiţiile tehnologiilor actuale, nu este recomandat să existe sateliţi mai apropiaţi de 2 grade în planul ecuatorial de 360 grade. La o spaţiere de 2 grade, pot exista pe cer, la un moment dat, doar 360/2=180 sateliţi de comunicaţie geostaţionari. Totuşi, fiecare transponder poate folosi mai multe frecvenţe şi polarizări pentru a creşte lungimea de bandă disponibilă.

Pentru a preveni un haos total pe cer, alocarea locurilor pe orbită este făcută de către ITU. Acest proces este mult influenţat de politică, existând ţări de abia ieşite din epoca de piatra care îşi cer locurile „lor” pe orbită (pentru a le închiria celui care oferă mai mult). Alte ţări susţin că drepturile naţionale de proprietate nu se extind până la lună, precum şi că nici o ţară nu are dreptul să revendice locurile de pe orbita de deasupra teritoriului său. Pentru a pune paie pe foc, telecomunicaţiile comerciale nu reprezintă singurul domeniu interesat de aceşti sateliţi. Canalele de televiziune, guvernele şi armata doresc şi ele câte o felie din „plăcinta” reprezentată de orbita terestra.

Sateliţii moderni pot fi destul de mari, cântărind până la 4000 de kg şi consumând câţiva KW de energie electrică produsă de panourile solare. Efectele forţelor de gravitaţie solare, lunare şi planetare tind să îi deplaseze din locurile şi orientările prevăzute pentru ei pe orbită, efect contracarat de motoarele de rachetă cu care sunt echipaţi. Această activitate de reglare fină se numeşte **menţinerea staţiei**. Cu toate acestea, când combustibilul pentru motoare se epuizează, de obicei în vreo 10 ani, satelitul pluteşte în derivă şi se roteşte neajutorat, astfel încât trebuie să fie scos din funcţionare. În cele din urma, orbita sa se deformează şi satelitul reintră în atmosfera unde este distrus prin ardere sau, uneori, se prăbuşeşte pe pământ.

Alocarea locurilor pe orbită nu este singurul măr al discordiei. Alocarea frecvenţelor este de asemenea un motiv de a se isca noi conflicte, deoarece transmisiile descendente de la sateliţi interferează cu utilizatorii dispozitivelor de comunicaţie prin microunde deja existente. Prin urmare, ITU a alocat anumite benzi de frecvenţă utilizatorilor comunicaţiilor prin sateliţi. Cele mai importante benzi comerciale sunt listate în fig. 2-16. Banda C a fost desemnată iniţial pentru traficul comercial prin sateliţi. În banda C sunt asigurate două domenii de frecvenţă, cea mai mică pentru traficul descendent (dinspre satelit), iar cea superioară pentru traficul ascendent (către satelit). Pentru o conexiune full-duplex este necesar un canal în ambele sensuri. Aceste benzi sunt deja supraaglomerate, deoarece sunt folosite şi de purtătoarele obişnuite pentru legăturile terestre pe microunde. Benzile L şi S au fost adăugate prin acordul internaţional din anul 2000. Oricum, şi ele sunt înguste şi aglomerate.

Banda	Legătura descendentă	Legătura ascendenta	Lăţime de bandă	Probleme
L	1.5 GHz	1.6 GHz	15 MHz	Lăţime mică de bandă; aglomerata
S	1.9 GHz	2.2 GHz	70 MHz	Lăţime mică de bandă; aglomerata
C	4.0 GHz	6.0 GHz	500 MHz	Interferenţe terestre
Ku	11 GHz	14 GHz	500 MHz	Ploaia
Ka	20 GHz	30 GHz	3500 MHz	Ploaia; costul echipamentelor

Fig. 2-16. Principalele benzi de satelit.

Următoarea bandă, cea mai înaltă, disponibilă pentru companiile comerciale de telecomunicaţii, este banda Ku (K under, rom: sub K). Această bandă nu este (încă) congestionată şi – la aceste frecvenţe – sateliţii pot fi poziţionaţi la o distanţă de un grad. Totuşi, există şi aici o altă problemă: ploaia. Apa este un absorbant excelent al acestor microunde scurte. Din fericire, furtunile torenţiale sunt de obicei localizate şi, prin urmare, folosind mai multe staţii terestre separate de distanţe mari (în loc de una singură), problema poate fi evitată cu preţul surplusului de antene, cabluri şi electronică necesare pentru a comuta rapid între staţii. Lăţimea de bandă pentru Ka (K above, rom: peste K) a fost şi ea alocată pentru traficul comercial prin satelit, dar echipamentele necesare pentru folosirea ei sunt încă foarte scumpe. În plus faţă de aceste benzi comerciale, există multe benzi guvernamentale şi militare.

Un satelit modern are în jur de 40 de transpondere, fiecare cu o lăţime de bandă de 80 MHz. Un transponder de 50 Mbps poate fi folosit pentru a codifica un singur flux de date de 50 Mbps, 800 canale vocale digitale de 64 Kbps sau diverse alte combinaţii. De obicei, fiecare transponder funcţionează ca un repetor, dar, mai nou, sateliţii au şi o anumita capacitate de procesare integrată, permiţând operaţii mai sofisticate. La sateliţii mai vechi, împărţirea transponderilor pe canale s-a făcut static, prin împărţirea lărgimii de bandă în benzi fixe de frecvenţă (FDM). În prezent, fiecare fascicol de transponder este împărţit în intervale de timp, cu mai mulţi utilizatori comunicând pe rând. Vom studia aceste două tehnici (multiplexarea cu divizare în frecvenţă şi multiplexarea cu divizare în timp) într-o secţiune următoare din acest capitol.

Primii sateliţi geostaţionari aveau un singur fascicol spaţial care ilumina aproximativ 1/3 din suprafaţa Pământului, suprafaţă numită **rază de acţiune**. Odată cu scăderea masivă a preţului, dimensiunii şi cerinţelor de putere ale microelectronicii, a devenit posibilă o strategie de difuzare mai complexă. Fiecare satelit este echipat cu antene şi transpondere multiple. Fiecare fascicol descendent poate fi focalizat pe o arie geografică mică şi prin urmare pot avea loc simultan, transmisii ascendente şi descendente multiple. Aceste aşa numite **fascicole punctuale** sunt în mod obişnuit de formă eliptică şi pot avea până la câteva sute de km în diametru. Un satelit de comunicaţii pentru Statele Unite are în mod normal un singur fascicol larg pentru cele 48 de state alăturate, plus fascicole punctuale pentru Alaska şi Hawaii.

O nouă realizare în lumea comunicaţiilor prin satelit o constituie dezvoltarea microstaţiilor de cost scăzut, denumite şi **VSAT**-uri (**Very Small Aperture Terminals**, rom: terminale cu deschidere foarte mică) (Abramson, 2000). Aceste mici terminale au antene de 1 metru sau mai mici (fata de cei 10m ai unei antene

GEO standard) şi pot emite cu o putere de aproximativ IWatt. Legătura ascendentă este în general bună pentru 19.2Kbps, dar cea descendentă este mai mare, deseori de 512 Kbps. Televiziunea cu difuzare directă prin satelit foloseşte această tehnologie pentru transmisia uni-direcţională.

În multe sisteme VSAT, microstaţiile nu au suficientă putere pentru a comunica direct între ele (prin intermediul satelitului, desigur). În schimb, se foloseşte o staţie terestră specială, un hub, cu o antenă mare, de câştig ridicat, pentru a retransmite traficul dintre VSAT-uri, aşa cum este prezentat în fig. 2-17. În acest fel, atât emiţătorul cât şi receptorul dispun de o antenă mare şi de un amplificator puternic. Compromisul constă într-o întârziere mai mare în schimbul unor staţii mai ieftine la utilizatorul final.

VSAT-urile au un potenţial ridicat în special pentru zonele rurale. Nu este un lucru foarte cunoscut, dar peste jumătate din populaţia globului locuieşte la peste o ora de mers pe jos de cel mai apropiat post telefonic. A instala fire telefonice până la fiecare dintre miile de sate mici este mult peste bugetele majorităţii guvernelor din lumea a treia, dar instalarea de antene VSAT de 1 metru alimentate de celule solare este o soluţie adeseori posibilă. VSAT-urile oferă o tehnologie care va conecta întreaga lume.

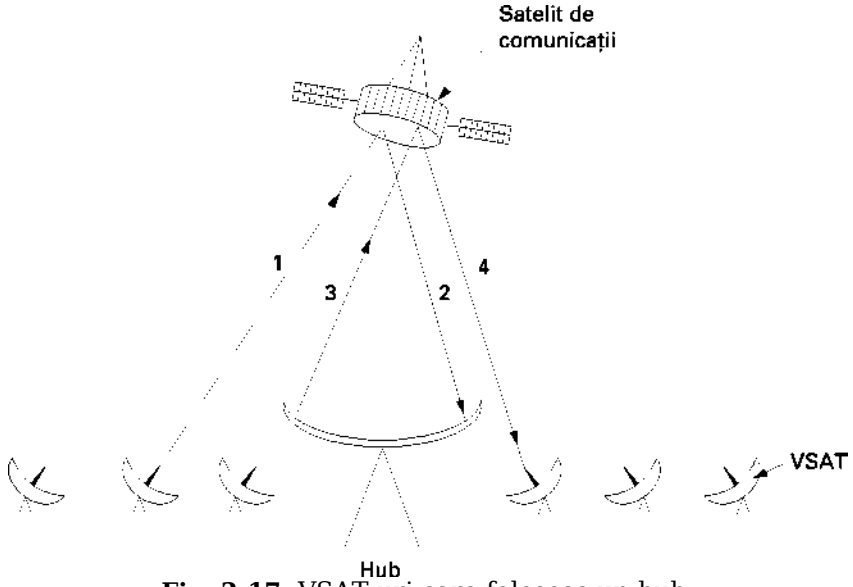


Fig. 2-17. VSAT-uri care folosesc un hub.

Sateliţii de comunicaţie au câteva proprietăţi prin care se deosebesc radical de legăturile terestre punct-la-punct. Ca un prim aspect, cu toate că semnalele spre şi dinspre satelit se propagă cu viteza luminii (aproximativ 300.000 km/sec), lungimea semnificativă a traseului dus-întors introduce o întârziere substanţială pentru sateliţii **GEO**. Funcţie de distanţa dintre utilizator şi staţia terestră, precum şi de înălţimea satelitului deasupra orizontului, timpul de propagare este între 250 şi 300 msec. O valoare uzuală este de 270 msec (540 msec pentru un sistem VSAT cu un hub).

Trebuie spus, pentru comparaţie, că legăturile terestre prin microunde au o

întârziere de propagare în jur de 3 psec/km, iar legăturile pe cablu coaxial sau fibră optică au o întârziere de aproximativ 5psec/km. Transmisia prin cel de-al doilea mediu este mai înceată decât prin primul deoarece semnalele electromagnetice se propagă mai repede prin aer decât prin materiale solide.

O altă proprietate importantă a sateliţilor este aceea că ei sunt în mod inerent sisteme cu difuzare. Transmiterea unui mesaj către miile de staţii din raza de acţiune a unui transponder costă tot atât de mult cât pentru o singură staţie. Pentru unele aplicaţii, această proprietate este foarte utilă. De exemplu, se poate imagina un satelit difuzând pagini Web populare către memoriile cache ale unui mare număr de computere răspândite pe o arie largă. Chiar şi atunci când difuzarea poate fi simulată folosind linii punct-la-punct, difuzarea prin satelit poate fi mult mai ieftină. Pe de altă parte, din punct de vedere al securităţii şi confidenţialităţii, sateliţii sunt un dezastru complet: oricine poate asculta orice. Criptarea este esenţială dacă securitatea este o necesitate.

Sateliţii au şi proprietatea că preţul transmisiei unui mesaj este independent de distanţa parcursă. Un apel peste ocean nu costă mai mult decât un apel peste stradă. Sateliţii au rate de eroare foarte scăzute şi pot fi instalaţi aproape instantaneu, un considerent major pentru comunicaţiile militare.

2.4.2 Sateliţi de altitudine medie

La altitudini mult mai joase, între cele două centuri Van Allen, găsim sateliţii MEO (MediumEarth Orbit, rom: orbită terestră medie). Văzuţi de pe Pământ, aceşti sateliţi se deplasează lent în plan longitudinal, iar un ocol în jurul Pământului durează cam de 6 ore. În consecinţă, ei trebuie urmăriţi în timp ce trec pe deasupra Pământului. Având altitudine mai mică decât **GEO**, au o rază de acţiune mai mică pe Pământ şi este nevoie de emiţătoare mai puţin puternice pentru a comunica cu ei. Deocamdată nu sunt folosiţi pentru comunicaţii, deci nu vor mai fi examinaţi aici. Cei 24 de sateliţi **GPS** (**Global Positioning System**, rom: sistem de poziţionare global) care orbitează la aproximativ 18.000 km sunt exemple de sateliţi MEO.

2.4.3 Sateliţi de joasă altitudine

Coborând în altitudine, ajungem la sateliţii **LEO** (**Low-Earth Orbit**, rom: orbită terestră joasă). Datorită mişcării lor rapide, este nevoie de mulţi astfel de sateliţi pentru a realiza un sistem complet. Pe de altă parte, datorită faptului că sateliţii sunt atât de aproape de Pământ, staţiile terestre nu au nevoie de multă putere, iar întârzierea dus-întors este numai de câteva milisecunde. În acest paragraf vom examina trei exemple, două concepute pentru comunicaţiile de voce şi unul pentru servicii Internet.

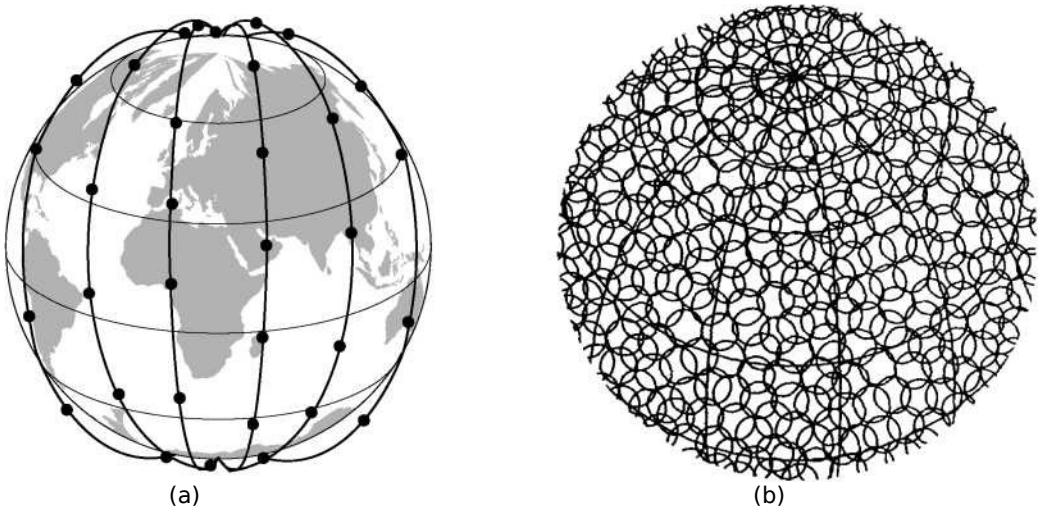
Iridium

Aşa cum am menţionat şi mai sus, în primii 30 de ani ai erei sateliţilor, sateliţii de joasă altitudine au fost rareori folosiţi, deoarece ei apar şi dispar destul de repede din câmpul vizual. În 1990, Motorola a declanşat o acţiune în acest domeniu prin punerea la punct a unei noi aplicaţii. Motorola a obţinut acordul FCC în vederea lansării a 77 de sateliţi de joasă altitudine pentru proiectul Iridium (elementul 77 este Iridium). Planul a fost mai târziu revăzut, astfel încât să se utilizeze numai 66

de sateliţi şi, ca urmare, proiectul ar fi trebuit să fie redenumit Dysprosium (elementul 66), dar probabil că sună prea mult ca o boală. Ideea era că în momentul în care un satelit dispare din câmpul vizual, un alt satelit ar putea să-i ia locul. Această propunere a generat printre celelalte companii de telefoane o poftă nebună: dintr-o dată, toată lumea dorea să lanseze un lanţ de sateliţi de joasă altitudine.

După şapte ani în care s-au adunat parteneri şi finanţări, sateliţii Iridium s-au lansat în 1997. Serviciile de comunicaţii au început în noiembrie 1998. Din păcate, cererea comercială pentru telefoane mari şi grele, chiar dacă acestea comunicau prin satelit, era neglijabilă deoarece reţeaua de telefonie mobilă crescuse spectaculos din 1990. În consecinţă, Iridium nu era profitabil şi a fost forţat să intre în faliment în august 1999, într-unul din cele mai spectaculoase fiasco-uri de corporaţii din istorie. În consecinţă, sateliţii şi celelalte bunuri (în valoare de 5 miliarde $) au fost cumpărate de un investitor cu 25 milioane $ într-un fel de vânzare cu reducere dintr-un garaj extraterestru. Serviciul Iridium a fost repornit în martie 2001.

Scopul principal al sistemului Iridium era (şi este) să furnizeze servicii mondiale de telecomunicaţie, folosind dispozitive portabile care să comunice direct cu sateliţii Iridium. Sistemul furnizează servicii vocale, de date, paging, fax şi navigare, oriunde pe pământ, apă şi aer. Printre clienţi se numără industriile maritimă, aviatică şi de explorare de petrol, ca şi persoanele care călătoresc în părţi ale lumii lipsite de o infrastructură de telecomunicaţii (de exemplu deşerturi, munţi, jungle şi unele ţări din lumea a treia).



(a) (b)

Fig. 2-18. (a) Sateliţii Iridium formează şase coliere în jurul Pământului. (b) 1628 de celule mişcătoare acoperă Pământul.

Sateliţii Iridium sunt poziţionaţi la o altitudine de 750 km pe orbite polare circulare. Ei sunt aranjaţi în formă de coliere nord-sud, cu un satelit la fiecare 32 grade latitudine. După cum se sugerează în fig. 2-18(a), cu şase coliere de sateliţi s-ar putea acoperi întregul Pământ. Cei care nu cunosc prea multe despre chimie, se pot gândi la această dispunere ca la un atom de dysprosium foarte mare, având Pământul pe post de nucleu şi sateliţii pe post de electroni.

După cum este prezentat în fig. 2-18(b), fiecare satelit va avea cel mult 48 de raze punctuale, cu un total de 1628 celule pe suprafaţa Pământului. Fiecare satelit are o capacitate de 3840 de canale, sau 253.440 în total. O parte din acestea sunt folosite pentru paging şi navigare, altele pentru date şi voce.

O proprietate interesantă a sistemului Iridium este aceea că o comunicaţie între clienţi aflaţi la distanţă are loc în spaţiu, cu un satelit transmiţând date la altul, cum este ilustrat în fig. 2-19(a). Aici putem vedea un apelant de la Polul Nord contactând un satelit aflat direct deasupra sa. Apelul este transmis prin ceilalţi sateliţi şi trimis în cele din urma la apelatul de la Polul Sud.

Globalstar

Un proiect alternativ pentru Iridium este Globalstar. Acesta se bazează pe 48 de sateliţi LEO, dar foloseşte o schemă de comutare diferită de cea folosită de Iridium. În timp ce Iridium transmite apeluri de la satelit la satelit, ceea ce necesita echipamente complicate de comutare în cadrul sateliţilor, Globalstar utilizează un sistem de repetoare tradiţional. Apelul pornit de la Polul Nord din fig. 2-19(b) este trimis înapoi către pământ şi preluat de staţia terestră mare de la Atelierul lui Moş Crăciun. Apelul este apoi dirijat printr-o reţea terestră către staţia terestră cea mai apropiată de apelat şi livrat printr-o conexiune cu satelitul repetor, aşa cum se vede şi în figură. Avantajul acestei scheme este că transferă mare parte din complexitate pe Pământ, unde este mai prelucrările sunt mai uşor de efectuat. De asemenea, dacă pentru staţiile terestre se vor folosi antene mari, care pot emite un semnal puternic şi pot recepţiona unul slab, atunci în sistem pot fi folosite telefoane cu putere mică. Până la urmă, telefonul debitează o putere de câţiva mW, deci semnalul care se întoarce la staţia terestră este destul de slab, chiar după ce a fost amplificat de satelit.

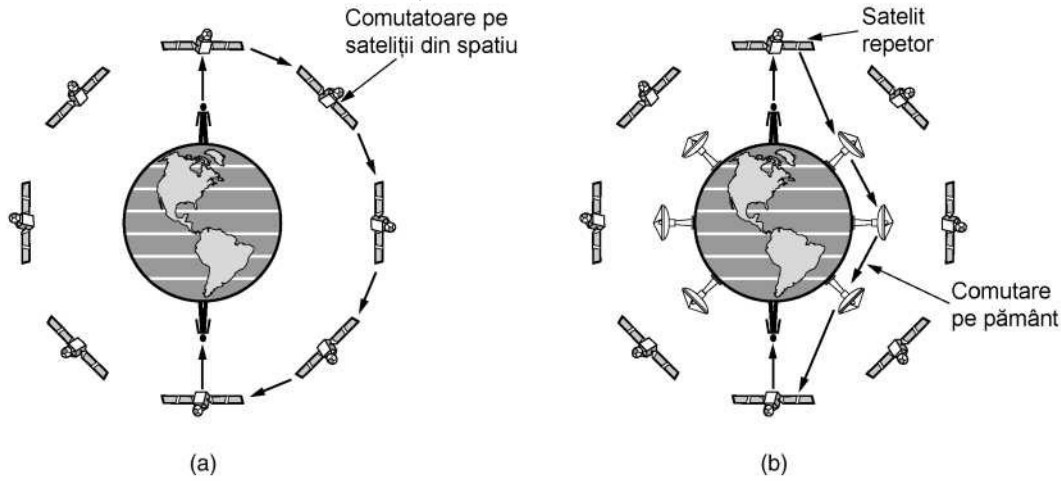


Fig. 2-19. (a) Transmisia în spaţiu. (b) Transmisia pe Pământ.

Teledesic

Iridium este conceput pentru utilizatorii de telefoane aflaţi în locuri ciudate. Următorul nostru exemplu, **Teledesic**, este conceput pentru utilizatorii de Internet din întreaga lume doritori de cât mai multă lărgime de bandă. A fost conceput în 1990 de către pionierul telefoniei mobile Craig McCaw şi de către fondatorul Microsoft, Bill Gates, care era nemulţumit de viteza de melc cu care companiile telefonice din lume ofereau lăţime ridicată de bandă utilizatorilor de calculatoare. Scopul sistemului Teledesic este să ofere milioanelor de utilizatori Internet care accesează sistemul simultan o legătură ascendentă de până la 100 Mbps şi o legătură descendentă de până la 720 Mbps, utilizând o antenă de tip VSAT mică şi fixă, dar şi ocolind total sistemul telefonic. Pentru companiile telefonice, acesta este un ideal de neatins.

Proiectul original consta dintr-un sistem format din 288 de sateliţi cu rază mică de acţiune, poziţionaţi în 12 planuri imediat sub centura Van Allen, la o altitudine de 1350 km. Acest proiect a fost schimbat ulterior la 30 de sateliţi cu raze de acţiune mai mari. Transmisia se petrece în banda Ka, relativ neaglomerata şi cu lăţime de bandă ridicată. Sistemul funcţionează cu comutare de pachete în spaţiu, fiecare satelit fiind capabil să dirijeze pachete către vecinii săi. Când un utilizator are nevoie de lăţime de bandă pentru a trimite pachete, aceasta este cerută şi alocată dinamic în aproximativ 50msec. Este prevăzut ca sistemul să intre în funcţiune în anul 2005, dacă totul merge conform planului.

2.4.4 Sateliţii în comparaţie cu fibrele optice

O comparaţie între comunicaţiile prin satelit şi comunicaţiile terestre este instructivă. Cu numai 20 de ani în urmă s-ar fi putut crede că viitorul va aparţine comunicaţiilor prin satelit. La urma urmei, sistemul telefonic s-a schimbat puţin în ultimii 100 de ani şi nici nu dă semne de schimbare în următorii 100 de ani. Această evoluţie lentă s-a datorat în mare măsură mediului înconjurător, în care companiilor

de telefoane li se cerea să furnizeze un serviciu vocal calitativ la un preţ rezonabil (ceea ce au şi făcut) în schimbul unui profit garantat al investiţiilor lor. Prin urmare, pentru cei care aveau date de transmis, erau disponibile modemuri de 1200 bps. Aceasta era destul de bine pentru ceea ce exista atunci.

Introducerea competiţiei în 1984 în Statele Unite şi, ceva mai târziu, în Europa a schimbat radical situaţia. Companiile de telefoane au început înlocuirea cu fibre optice a reţelelor exploatate un timp atât de îndelungat şi au introdus servicii cu lăţimi de bandă ridicate, cum este ADSL (Asymetric Digital Subscriber Line, rom: linie de abonat digitală asimetrica). În plus, şi-au încetat practica îndelungată de a factura utilizatorii aflaţi la distanţe mai mari cu sume mărite artificial pentru a subvenţiona utilizatorii locali.

Dintr-o dată, se părea că legăturile terestre pe fibră optică reprezintă câştigătorul pe termen lung. Cu toate acestea, sateliţii de comunicaţie au câteva nişe pe piaţă, în care fibra optică nu a pătruns (şi în unele cazuri nici nu o va putea face). Vom analiza acum o parte dintre acestea.

În timp ce o singură fibră optică are, în principiu, mai multă lăţime potenţială de bandă decât toţi sateliţii lansaţi vreodată, această lăţime de bandă nu este disponibilă majorităţii utilizatorilor. Fibrele optice instalate la ora actuală sunt folosite în sistemul telefonic pentru a gestiona simultan mai multe apeluri de distanţă lungă, şi nu pentru a furniza utilizatorilor individuali lăţime de bandă ridicată. În cazul sateliţilor, un utilizator poate foarte bine să scoată o antenă pe acoperişul clădirii şi să ocolească sistemul telefonic. Teledesic se bazează pe această idee.

O a doua nişă o reprezintă comunicaţiile mobile. În zilele noastre, mulţi oameni doresc să comunice în timp ce fac jogging, conduc, navighează sau zboară. Legăturile terestre prin fibre optice nu le sunt de nici un folos, în schimb le pot fi utile legăturile prin satelit. Este posibil, totuşi, ca o combinaţie între radioul celular şi fibra optică să fie satisfăcătoare pentru cerinţele majorităţii utilizatorilor (probabil cu excepţia acelora care se află la bordul unui avion sau pe mare).

O a treia nişă o reprezintă situaţiile în care este esenţială difuzarea. Un mesaj transmis de satelit poate fi recepţionat simultan de mii de staţii terestre. De exemplu, o firmă care transmite acţiuni, titluri de proprietate sau preţurile mărfurilor către mii de distribuitori, ar putea descoperi că utilizarea unui sistem prin satelit este mult mai ieftină decât simularea difuzării pe Pământ.

O a patra nişă o constituie comunicaţia în locurile cu terenuri greu accesibile sau cu o infrastructură terestră slab dezvoltată. Indonezia, de exemplu, are propriul satelit pentru traficul telefonic intern. Lansarea unui satelit a fost mult mai simplă decât întinderea a mii de cabluri submarine între cele 13.677 de insule din arhipelag.

O a cincea nişă pentru piaţa sateliţilor este acolo unde dreptul de instalare a fibrei optice este dificil de obţinut sau nejustificat de scump.

În al şaselea rând, atunci când instalarea rapidă este critică, cum este în cazul sistemelor de comunicaţii militare pe timp de război, sateliţii obţin câştig de cauză fără probleme.

Pe scurt, se pare că în viitor fluxul principal de comunicaţie va fi pe fibră optică în combinaţie cu radioul celular, iar pentru câţiva utilizatori specializaţi, vor rămâne

preferabili sateliţii. Totuşi, există un avertisment valabil pentru toate acestea: economia. Cu toate că fibra optică oferă mai multă lăţime de bandă, este posibil, fără îndoială, ca în viitor comunicaţiile terestre şi cele prin satelit să intre într-o competiţie agresivă pe baza preţului practicat. Dacă progresele tehnologice vor reduce radical costul de instalare al unui satelit (de exemplu, unele viitoare navete spaţiale vor putea împrăştia în spaţiu mai multe zeci de sateliţi la o singură lansare) sau dacă vor deveni populari sateliţii de joasă altitudine, atunci s-ar putea ca fibrele optice să-şi piardă, pe unele pieţe, poziţia lor de lider.

2.5 SISTEMUL TELEFONIC

Două calculatoare ale aceleiaşi companii sau organizaţii, aflate la mică distanţă, pot fi conectate simplu printr-un cablu, pentru a comunica între ele. Acesta este modul de funcţionare al reţelelor locale. Oricum, când distanţele sunt mari sau sunt multe calculatoare, ori când cablurile ar trebui să treacă printr-un loc public, costul instalării de cabluri particulare este de obicei prohibitiv. Mai mult, în aproape toate ţările din lume, instalarea de cabluri de-a lungul (sau pe sub) proprietăţile publice este ilegală. În consecinţă, proiectanţii de reţele trebuie să se bazeze pe facilităţile de comunicaţie existente.

O astfel de facilitate este **PSTN**, (**Public Switched Telephone Network**, rom: reţea telefonică publică comutată), care a fost proiectată cu mulţi ani în urmă, în cu totul alt scop: transmisia vocii umane într-o formă mai mult sau mai puţin recognoscibilă. Acest sistem nu este destul de potrivit pentru comunicaţiile între calculatoare, dar situaţia se schimbă rapid odată cu introducerea fibrelor optice şi a tehnologiei digitale. În orice caz, sistemul telefonic este atât de strâns legat de reţelele de calculatoare (larg răspândite geografic), încât merită să îi acordăm mai multă atenţie.

Pentru avea o idee despre ordinul de mărime al problemei, să facem o comparaţie scurtă dar semnificativă între proprietăţile unei conexiuni tipice între calculatoare printr-un cablu local şi printr-o linie telefonică. Un cablu care face legătura între două calculatoare poate transfera date la 10^9 bps, poate şi mai mult. Prin contrast, o linie telefonică are o viteza maximă de transfer de date de 56 Kbps, o viteză de aproape 20.000 de ori mai mică. Aceasta este diferenţa dintre o raţă care merge legănându-se în tihnă prin iarba şi o rachetă care zboară spre lună. Dacă linia telefonică este înlocuită de o conexiune ADSL, viteza este mai mică de 1000-2000 de ori.

Desigur, proiectanţii sistemelor de calculatoare cheltuiesc mult timp şi efort pentru a analiza cum pot fi acestea folosite cât mai eficient şi au dificultăţi cu un sistem a cărui performanţă (din punctul lor de vedere) este cu 3 sau 4 ordine de mărime mai slabă. În paragrafele care urmează vom descrie sistemele telefonice şi vom prezenta istoria şi viitorul lor. Pentru informaţii suplimentare despre structura sistemelor telefonice vezi (Bellamy 1991).

2.5.1 Structura sistemului telefonic

Curând după ce Alexander Graham Bell a brevetat telefonul în 1876 (doar cu

câteva ore înaintea rivalului său, Elisha Gray), cererea pentru noua sa invenţie a fost imensă. Piaţa iniţială consta în vânzarea telefoanelor, existente numai sub formă de perechi. Era la latitudinea clientului să întindă un fir între ele. Dacă proprietarul unui telefon dorea să comunice cu alţi *n* proprietari de telefoane, trebuiau folosite fire separate pentru conectarea tuturor celor *n* case. În mai puţin de un an, oraşele erau acoperite cu fire care treceau peste case şi copaci într-o încrengătură sălbatică. A devenit imediat evident că modelul conectării fiecărui telefon la fiecare alt telefon, ca în fig. 2-14(a), nu va putea funcţiona.

Bell a observat acest lucru şi a înfiinţat Bell Telephone Company, companie care a deschis primul oficiu de comutare (în New Haven, Connecticut), în 1878. Compania a întins câte un fir către casa sau biroul fiecărui client. Pentru a da un telefon, clientul lovea furca, generând astfel un semnal sonor în centrală, care atrăgea atenţia operatorului; acesta conecta apoi manual cei doi clienţi cu ajutorul unui cablu. Modelul unui oficiu de comutare este ilustrat în fig. 2-20(b).

Destul de repede, oficiile de comutare Bell Systems au apărut peste tot şi oamenii au simţit nevoia unor convorbiri interurbane, oficiile Bell System începând să se conecteze între ele. Problema iniţială a redevenit actuală: conectarea fiecărui oficiu de comutare cu fiecare alt oficiu prin intermediul unui cablu a scăpat rapid de sub control, şi astfel s-au inventat oficiile de comutare de nivelul doi. După un timp, au fost necesare mai multe oficii de nivelul doi, ca în fig. 2-20(c). În cele din urmă, ierarhia a ajuns până la 5 niveluri.

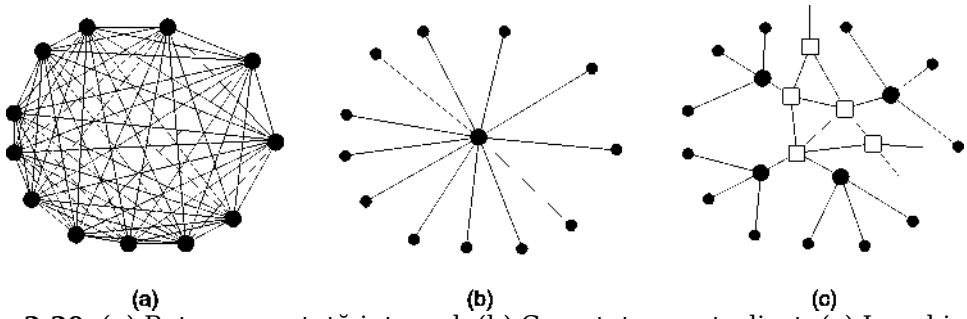


Fig. 2-20. (a) Reţea conectată integral. (b) Comutator centralizat. (c) Ierarhie pe două niveluri.

În 1890, cele trei componente majore ale sistemului telefonic erau puse la punct: oficiile de comutare, cablurile între clienţi şi oficiile de comutare (acum echilibrate, izolate, cablu torsadat în locul firelor neizolate, legate la pământ) şi legăturile între oficiile de comutare pe distanţă lungă. Cu toate că au apărut îmbunătăţiri în toate cele trei domenii, modelul de bază al sistemului Bell a rămas esenţialmente intact mai bine de 100 de ani. Pentru o scurtă istorie a sistemului telefonic, vezi (Hawley, 1991).

Anterior pătrunderii în forţă pe piaţă a companiei AT&T în 1984, sistemul telefonic era implementat ca o ierarhie pe multe niveluri cu un grad mare de redundanţă. Descrierea care urmează, deşi foarte simplificată, conţine totuşi esenţialul. Fiecare telefon are două fire de cupru conectate direct la cel mai

apropriat **oficiu final** (deseori numit **oficiu central local)**. Distanţa este în mod uzual între 1 şi 10 Km, fiind mai mică în oraşe decât în zonele rurale. Numai în Statele Unite sunt peste 19.000 de oficii finale. Concatenarea codului zonei şi a primelor trei cifre din numărul de telefon specifică în mod unic un oficiu final. Legătura formată de cele două fire între un telefon şi oficiul final corespunzător este cunoscută în termeni tehnici sub numele de **buclă locală**. Dacă toate buclele locale din toată lumea ar fi fost puse cap la cap, ele ar acoperi distanţa de la pământ la Lună şi înapoi de 1000 de ori.

La un moment dat, 80 la sută din capitalul AT&T era constituit de cuprul din buclele locale. AT&T era atunci, de fapt, cea mai mare mină de cupru din lume. Din fericire, acest lucru nu era prea mult cunoscut în lumea investiţiilor. Dacă s-ar fi cunoscut, AT&T putea fi cumpărată, lichidate toate serviciile telefonice din Statele Unite, smuls tot cablul şi vândut unei rafinării de cupru pentru un profit imediat.

Dacă un abonat ataşat la un anumit oficiu final apelează alt abonat ataşat la acelaşi oficiu final, mecanismul de comutare din acel oficiu stabileşte o legătură electrică directă între cele două bucle locale. Această legătură rămâne intactă pe toată durata convorbirii.

Dacă telefonul apelat este ataşat la un alt oficiu final, trebuie folosită o altă procedură. Fiecare oficiu final are un număr de linii conectate la unul sau mai multe centre de comutare apropriate, numite **oficii de taxare** (sau, dacă sunt în aceeaşi zonă, **oficii în tandem)**. Aceste linii se numesc **trunchiuri de conectare la oficiile de taxare** (**toll connecting trunks**). Dacă se întâmplă ca oficiul final al celui care apelează şi oficiul celui apelat să aibă trunchi de conectare către acelaşi oficiu de taxare (ceea ce este probabil dacă sunt relativ apropriate), legătura poate fi stabilită de către oficiul de taxare. O reţea telefonică formată din telefoane (punctele mici), oficii finale (punctele mari) şi oficiile de taxare (pătratele) este prezentată în fig. 2-20(c).

Dacă apelantul şi apelatul nu au un oficiu de taxare în comun, calea va trebui să fie stabilită undeva mai sus în ierarhie. Oficiile de taxare sunt conectate prin intermediul unei reţele formate din oficii primare, sectoriale şi regionale. Comunicaţiile între oficiile de taxare primare, sectoriale şi regionale se realizează prin intermediul **trunchiurilor de comunicaţie** de bandă foarte largă (numite şi **trunchiuri de comunicaţie inter-oficii**). Varietatea centrelor de comutare şi a topologiei acestora (pot două oficii sectoriale să fie conectate direct sau prin intermediul unui oficiu regional ?) diferă de la o ţară la alta în funcţie de densitatea telefonică. Fig. 2-21 prezintă un mod posibil de a realiza o legătură pe distanţe medii. În telecomunicaţii sunt folosite diverse medii de transmisie. În prezent, buclele locale constau din cabluri torsadate de categoria 3 – cu toate că în primele zile ale telefoniei erau uzuale firele neizolate – aflate la o distanţa de 25 cm între ele, la polii telefonului. Între oficiile de comutare sunt larg folosite cablurile coaxiale, microundele şi mai ales fibrele optice.

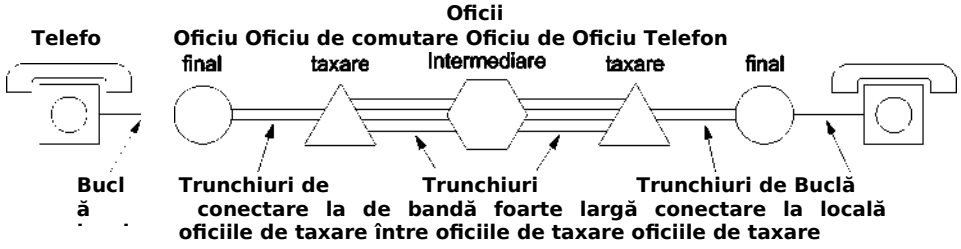


Fig. 2-21. Circuitul unei rute uzuale pentru o convorbire la distanţă medie

În trecut, transmisia în sistemul telefonic era analogică, semnalul vocal fiind transmis de la sursă la destinaţie sub forma unei tensiuni electrice. Odată cu apariţia fibrei optice, a electronicii digitale şi a calculatoarelor, toate trunchiurile şi comutatoarele sunt acum digitale, bucla locala rămânând ultima parte de tehnologie analogică a sistemului. Transmisia digitală este preferată, nefiind necesar să se reproducă exact o formă de undă analogică după ce a trecut prin mai multe amplificatoare într-o convorbire la distanţă. Distincţia între 1 şi 0 este suficientă. Această proprietate face ca transmisia digitală să fie de mai mare încredere decât cea analogică. De asemenea, este mai ieftină şi mai uşor de întreţinut.

Pe scurt, sistemul telefonic constă din trei componente majore:

1. Bucle locale (cablu torsadat analogic tras în interiorul caselor şi al companiilor).

2. Trunchiuri (fibre optice digitale care conectează oficiile de comutare).
3. Oficii de comutare (unde apelurile sunt transferate dintr-un trunchi în altul).

După o scurtă prezentare a politicii în domeniul telefonic, vom studia aceste trei componente mai în detaliu. Buclele locale oferă tuturor acces la întregul sistem, deci sunt critice. Din păcate, ele reprezintă şi veriga cea mai slabă a sistemului. Pentru trunchiurile pe distanţe mari, problema principală va fi gruparea mai multor convorbiri împreună şi trimiterea lor simultană. Soluţia se numeşte multiplexare şi vom studia trei modalităţi diferite de multiplexare. În sfârşit, există două moduri fundamental diferite de a face comutarea, aşa că le vom studia pe amândouă.

2.5.2 Politica din domeniul telefonic

Timp de mai multe zeci de ani, până în 1984, Bell Systems a asigurat servicii, atât pe distanţe scurte cât şi pe distanţe lungi, pentru aproape toată suprafaţa Statelor Unite. În anii 70, guvernul S.U.A. a ajuns la concluzia că acesta era un monopol ilegal şi a hotărât să îl anuleze. Guvernul a câştigat pe 1 ianuarie 1984, AT&T fiind destrămată în AT&T Long Lines, 23 de companii **BOC** (**Bell Operating Companies**) şi alte câteva părţi. Cele 23 de companii BOC erau grupate în şapte BOC regionale (RBOC) pentru a le face viabile din punct de vedere economic. Întreaga natură a telecomunicaţiilor în Statele Unite a fost schimbată peste noapte de o hotărâre judecătorească *(nu* de un act al Congresului).

Detaliile exacte ale acestei privaţiuni sunt descrise în aşa numita **MFJ** (**Modified Final Judgement**, rom: hotărâre finală modificată, un bun exemplu de oximoron*****, dacă a existat vreodată vreunul – dacă hotărârea putea fi modificată, este evident că nu era finală). Acest eveniment a condus la o creştere a competiţiei, la asigurarea unor servicii mai bune şi la scăderea preţurilor pentru clienţi. Totuşi, preţurile pentru serviciile locale au crescut şi subvenţiile încrucişate de la apelurile de lungă distanţa au fost eliminate, iar serviciile locale au trebuit să se susţină singure. Multe alte ţări iau în considerare introducerea competiţiei după acelaşi model.

Să clarificăm cum s-a putut atinge acest scop: Statele Unite au fost împărţite în aproape 160 de **LATA** (**Local Access and Transport Areas**, rom: zone de acces şi transport local). Pe scurt, o LATA acoperă o suprafaţă echivalentă ca dimensiuni cu o regiune acoperită de un acelaşi cod zonal. De obicei, în cadrul unei LATA există un **LEC** (**Local Exchange Carrier**, rom: transportator local), care deţine monopolul pentru un serviciu telefonic tradiţional din interiorul regiunii sale. Cele mai importante LEC sunt BOC-urile, cu toate că unele LATA conţin una sau mai multe din cele peste 1500 de companii telefonice independente care funcţionează ca LEC-uri.

Tot traficul inter-LATA este asigurat de un alt tip de companie, **IXC** (**IntereXchange Carrier**, rom: transportator inter-oficii). Iniţial, AT&T Long Lines era singura companie IXC serioasă, dar acum WorldCom şi Sprint sunt competitori consacraţi în domeniu. Una dintre preocupările apărute la separarea companiilor a fost ca toate IXC să fie tratate egal în ce priveşte calitatea liniilor, tarifele impuse şi numărul de cifre pe care un client trebuie să le formeze pentru a telefona. Modul în care acest lucru a fost îndeplinit este prezentat în fig. 2-22. Aici vedem trei exemple

***** Oximoron = contradicţie evidentă între termenii expresiei (n.t.)

de LATA, fiecare cu mai multe oficii finale. LATA-urile 2 şi 3 au de asemenea o mică ierarhie formată din oficii tandem (oficii intra-LATA).

Orice IXC care doreşte să asigure convorbiri provenite dintr-o LATA poate construi un oficiu de comutare, numit **POP** (**Point of Presence**, rom: punct de livrare). LEC-ul trebuie să conecteze fiecare IXC la fiecare oficiu final, direct, ca în LATA 1 şi 3, sau indirect, ca în LATA 2. Mai mult, condiţiile în care se face această conectare, atât tehnice cât şi financiare, trebuie să fie aceleaşi pentru toate IXC-urile. În acest mod, un abonat din LATA 1, să zicem, poate alege ce IXC să folosească pentru a apela abonaţi din LATA 3.

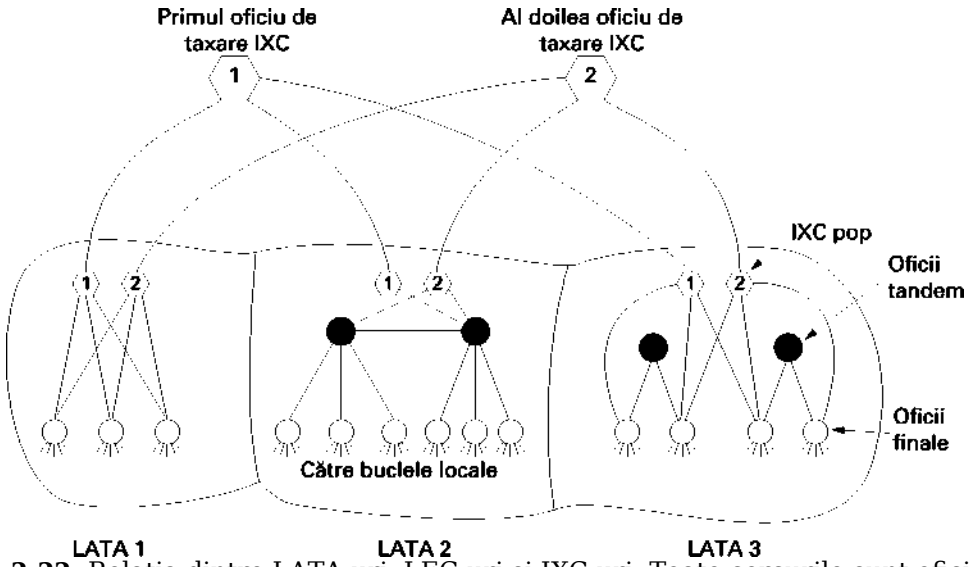


Fig. 2-22. Relaţia dintre LATA-uri, LEC-uri şi IXC-uri. Toate cercurile sunt oficii de comutare LEC. Fiecare hexagon aparţine câte unui IXC.

Printr-o clauză a MFJ, companiilor IXC le era interzis să ofere servicii telefonice locale şi tuturor LEC le era interzis să ofere servicii telefonice inter-LATA, cu toate că amândouă erau libere să intre în orice altă afacere, cum ar fi chiar deschiderea de restaurante. În 1984, această declaraţie era lipsită de ambiguităţi. Din păcate, tehnologia are un fel de a face ca legea să pară depăşită. Nici televiziunea prin cablu şi nici telefoanele celulare nu erau acoperite de această înţelegere. Atunci când televiziunea prin cablu a trecut de la forma unidirecţională la cea bidirecţională şi când a crescut brusc popularitatea telefoanelor celulare, atât LEC-urile cât şi IXC-urile au început să cumpere sau să se asocieze cu companiile din aceste domenii.

În 1995, Congresul a observat că menţinerea unei deosebiri între diferite tipuri de companii nu mai era posibilă şi a aprobat o notă prin care permitea companiilor de cablu TV, companiilor telefonice locale, transportatorilor pe distanţe mari şi operatorilor celulari să facă afaceri unul în domeniul celuilalt. Ideea era ca orice companie să poată oferi clienţilor săi un serviciu complet, cuprinzând cablul TV, telefon şi servicii informaţionale şi că diferite companii ar putea concura în ceea ce priveşte calitatea serviciilor asigurate şi preţul acestora. Această notă a fost legiferată în februarie 1996. În urma acestei hotărâri, unele BOC au devenit IXC şi

alte companii, precum operatorii de televiziune prin cablu, au început să ofere servicii telefonice locale în competiţie cu LEC-urile.

O proprietate interesantă a legii din 1996 este cerinţa ca LEC-urile să implementeze portabilita- tea locală a numerelor. Aceasta înseamnă ca un client poate schimba companiile locale de telefoane fără să trebuiască să obţină un nou număr de telefon. Aceasta prevedere scuteşte mulţi clienţi de o problemă serioasă şi îi poate ajuta în decizia de a schimba LEC-ul, ceea ce conduce la creşterea competiţiei. Ca rezultat, peisajul telecomunicaţiilor din SUA trece printr-o restructurare radicală. Din nou, multe alte ţări încep să urmeze acest model. Deseori, celelalte ţări aşteaptă să vad ă cum funcţionează un experiment în SUA. Dacă funcţionează bine, fac acelaşi lucru; dacă funcţionează râu, încearcă altceva.

2.5.3 Bucla locală: Modemuri, ADSL şi transmisia fără fir

Este timpul să pornim studiul nostru detaliat despre funcţionarea sistemului telefonic. Principalele părţi ale sistemului sunt ilustrate în figura2-23. Aici se văd buclele locale, trunchiurile, oficiile de taxare şi oficiile finale, ambele conţinând echipamente de comutare care comută apelurile. Un oficiu final are până la 10.000 de bucle locale (în SUA şi alte ţări mari). De fapt, până de curând, codul şi prefixul de zonă indicau oficiul final, astfel încât numărul (212) 601-xxxx aparţinea unui anumit oficiu final cu 10.000 de abonaţi, numerotaţi de la 0000 la 9999. Odată cu evoluţia competiţiei pentru serviciile locale, acest sistem nu mai era viabil, deoarece mai multe companii doreau să aibă codul oficiului final. De asemenea, numărul de coduri era practic epuizat, astfel încât au trebuit introduse scheme de organizare mai complexe.

Să începem cu partea cu care cei mai mulţi oameni sunt familiarizaţi: bucla locală, formată din două fire care vin de la un oficiu final al unei companii telefonice şi intră în case sau companii mai mici. Bucla locală mai este numită adesea şi „ultima milă”, deşi lungimea ei poate fi de până la câteva mile. În ultimii 100 de ani, bucla locală a folosit semnalizarea analogică şi probabil va continua să o folosească timp de ani buni, datorită costului mare al conversiei la digital. Totuşi, până şi în acest ultim bastion al transmisiei analogice au loc schimbări. În acest capitol, vom studia bucla locală tradiţională şi noile îmbunătăţiri care au loc în acest domeniu, concentrându-ne pe comunicaţia de date de la calculatoarele casnice.

Atunci când un calculator doreşte să trimită date numerice pe o linie telefonică, datele trebuie să fie convertite în prealabil în formă analogică pentru a putea fi transmise pe o buclă locală . Aceasta conversie este făcută de către un modem, dispozitiv pe care îl vom studia în curând. La oficiul final al companiei telefonice, aceste date sunt convertite la forma digitală pentru a fi transmise pe trunchiurile pentru distanţe mari.

Dacă la celălalt capăt se află un calculator cu un modem, este necesara conversia inversă – digital la analogic pentru a putea traversa bucla locală către destinaţie. Această schemă este prezentată în fig. 2-23 pentru ISP-ul 1 (Internet Service Provider, rom: furnizor de servicii internet), care dispune de o bancă de modem-uri, fiecare fiind conectat la o altă buclă locală. Acest ISP poate servi atâtea conexiuni câte modemuri are (presupunând că serverul sau serverele sale au destulă putere de calcul). Această schemă era considerată normală până când au apărut modem-urile

de 56 Kbps, din motive care se vor vedea în curând.

Codificarea analogică a semnalului constă în modificarea tensiunii electrice în funcţie de timp, pentru de a reprezenta un şir de date. Dacă mediul de transmisie ar fi fost ideal, receptorul ar fi primit exact acelaşi semnal pe care l-a expediat transmiţătorul. Din păcate, mediile nu sunt perfecte, iar semnalul recepţionat nu este identic cu semnalul transmis. Pentru datele numerice, aceste diferenţe pot conduce la erori.

Pe liniile de transmisie apar trei mari probleme: atenuarea, distorsiunea datorată întârzierii şi zgomotul. **Atenuarea** reprezintă pierderea de energie în timpul propagării semnalului. Pierderea se exprimă în decibeli pe kilometru. Energia pierdută depinde de frecvenţa semnalului. Pentru a vizualiza efectul acestei dependenţe de frecvenţă, să ne imaginăm un semnal nu ca o simplă undă, ci sub forma unei serii de componente Fourier. Fiecare componentă este atenuată diferit, ceea ce are ca rezultat la receptor un spectru Fourier diferit.

Pentru a agrava situaţia, diferitele componente Fourier se propagă cu viteze diferite de-a lungul firului. Aceste diferenţe de viteză duc la **distorsionarea** semnalului recepţionat la celălalt capăt.

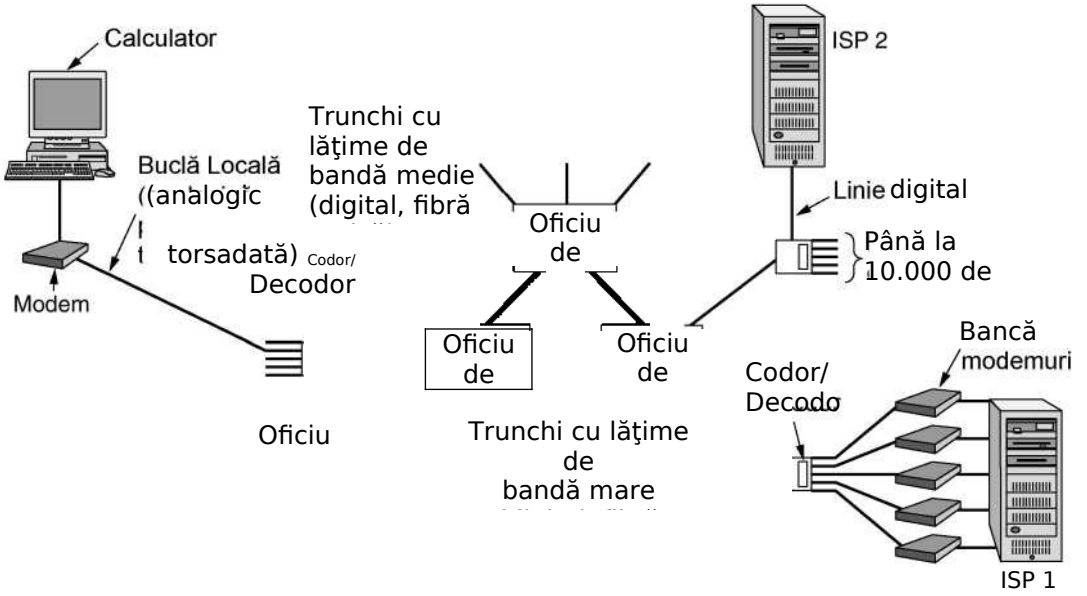


Fig. 2-23. Conectarea între calculatoare foloseşte transmisia analogică şi cea digitală.
Conversia este realizată de către modemuri şi de către codoare/decodoare.

O altă problemă este **zgomotul**, care reprezintă energie nedorită, provenită din alte surse decât transmiţătorul. Zgomotul termic este cauzat de mişcarea aleatorie a electronilor printr-o sârmă şi nu se poate evita. Interferenţa este produsă de cuplajul inductiv care se formează între două fire care sunt apropriate unul de altul. Atunci când vorbim la telefon, putem auzi o altă conversaţie în fundal. Aceasta este interferenţa. În sfârşit, există şi zgomote de tip impuls, determinate de şocuri electrice sau de alte cauze. Pentru datele digitale, zgomotele de tip impuls pot duce la dispariţia unuia sau a mai multor biţi.

Modemurile

Datorită problemelor prezentate anterior, în special datorită faptului că atât atenuarea cât şi viteza de propagare sunt dependente de frecvenţă, se doreşte evitarea prezenţei unui domeniu larg de frecvenţe într-un semnal.

Din păcate, undele pătratice, precum cele din datele numerice, au un spectru larg şi, în concluzie, suferă o atenuare puternică şi distorsiuni de întârziere. Aceste efecte fac din codificarea analogică în bandă de bază (DC) o alegere nepotrivită, cu excepţia situaţiilor în care se utilizează viteze mici şi transmisia are loc pe distanţe scurte.

Pentru a evita problemele asociate cu codificarea analogică în bandă de bază (DC), în special pe liniile telefonice, se utilizează codificarea analogică AC. Se introduce un ton continuu în domeniul 1000 – 2000 de Hz, numit **undă purtătoare sinusoidală**. Amplitudinea, frecvenţa sau faza acestei unde pot fi modulate. În **modularea în amplitudine**, sunt folosite două niveluri de tensiune pentru a reprezenta 0 şi 1, respectiv. În **modularea în frecvenţă**, cunoscută de asemenea sub denumirea de **codare prin deplasarea frecvenţei** (**frequency shift keying**), se

folosesc două (sau mai multe) tonuri diferite. (Termenul de **codare** este larg folosit în industrie ca sinonim pentru modulare). În varianta cea mai simplă, cea a **modulării în fază**, unda purtătoare este sistematic comutată la intervale egale

la 45, 135, 225, sau 315 grade. Fiecare schimbare de fază transmite 2 biţi de informaţie. De asemenea, obligativitatea unei schimbări de fază la sfârşitul fiecărui interval face receptorul să recunoască mai uşor limitele intervalelor de timp.

Fig. 2-24 ilustrează cele trei forme de modulare. În figura2-24(a) una dintre amplitudini este diferită de zero şi una este zero. În fig. 2-24(b) sunt folosite două frecvenţe. În fig. 2-24(c) o deplasare de fază este sau nu prezentă la marginile fiecărui bit. Un echipament care acceptă un şir serial de biţi la intrare şi produce o purtătoare modulată la ieşire (sau vice-versa) se numeşte **modem** (modulator-demodulator). Modemul este inserat între calculator (digital) şi sistemul telefonic (analogic).

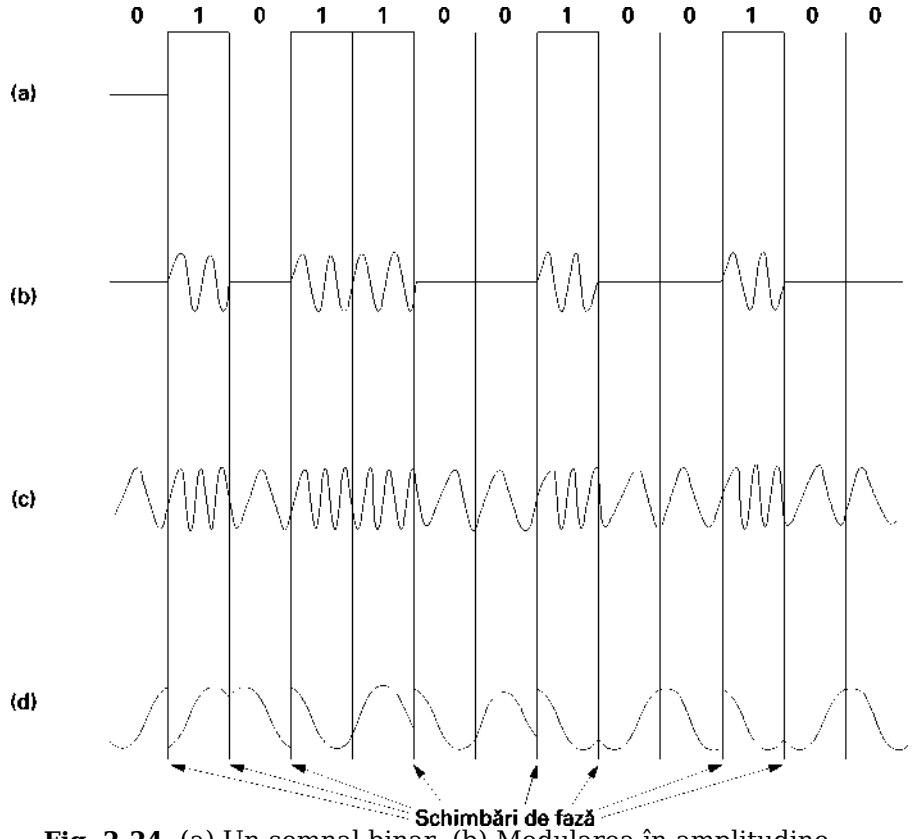


Fig. 2-24. (a) Un semnal binar. (b) Modularea în amplitudine. (c) Modularea în frecvenţă. (d) Modularea în fază.

Atingerea unor viteze din ce în ce mai mari nu este posibilă doar prin creşterea continuă a ratei de eşantionare. Teorema lui Nyquist afirmă că eşantionarea la o frecvenţa mai mare de 6000 de Hz este lipsită de interes chiar şi pentru o linie ideală de 3000 de Hz (nu este nici pe departe cazul unei linii telefonice). În practică, majoritatea modem-urilor eşantionează de 2400 ori/sec şi se concentrează să transmită cât mai mulţi biţi pe eşantion.

Numărul de eşantioane pe secundă se măsoară în **baud**. Pe durata fiecărui baud

este trimis un simbol. Astfel, o linie de *n* baud trimite *n* simboluri/sec. De exemplu, o linie de 2400 baud trimite un simbol la aproximativ fiecare 416,667psec. Dacă simbolul codifică prin tensiune nulă un bit 0 logic şi prin tensiunea de 1V un bit 1 logic, rata de bit este de 2400 bps. Dacă însă sunt folosite tensiunile de 0, 1, 2 şi 3 volţi, fiecare simbol codifică 2 biţi, astfel că o linie de 2400 baud poate transmite 2400 sim- boluri/sec la o rată de date de 4800 bps. În mod similar, pentru patru deplasări de fază posibile, sunt codificaţi tot 2 biţi/simbol, deci avem din nou o rată de bit dublă faţă de viteza de transmisie a liniei. Cea de-a doua tehnică este larg folosită şi se numeşte **QPSK** (**Quadrature Phase Shift Keying**, rom: modulaţia cuadratică în fază).

Conceptele de lărgime de bandă, viteza de transmisie (eng: baudrate), rată de simboluri şi rată de biţi sunt adeseori confundate, deci le vom reformula aici. Lărgimea de bandă a unui mediu reprezintă spectrul de frecvenţe care trec prin el cu atenuare minima. Este o proprietate fizică a mediului (de obicei de la 0 la o frecvenţă maximă) şi se măsoară în Hz. Viteza de transmisie reprezintă numărul de eşantioane preluate într-o secundă. Prin fiecare eşantion se transmite o parte din informaţie, adică un simbol. Deci, viteza de transmisie şi rata de simboluri sunt unul şi acelaşi lucru. Tehnica de modulare (de exemplu QPSK) determină numărul de biţi/simbol. Rata de biţi reprezintă cantitatea de informaţie trimisă prin canal şi este egală cu numărul de simboluri/sec înmulţit cu numărul de biţi/simbol.

Toate modemurile performante folosesc o combinaţie de tehnici de modulare pentru a transmite mai mulţi biţi pe baud. Deseori mai multe amplitudini şi mai multe deplasări de faza sunt combinate pentru a transmite mai mulţi biţi/simbol. În fig. 2-25(a), vedem puncte la 45, 135, 225 şi 315 grade, cu amplitudine constanta (distanţa faţă de origine). Faza unui punct este indicata de unghiul pe care lar face axa x cu o linie care uneşte punctul cu originea. Fig. 2-25(a) are patru combinaţii posibile şi poate fi folosită pentru a transmite 2 biţi pe simbol. Este exact QPSK.

În fig. 2-25(b) vedem o schemă diferită de modulare, în care sunt folosite 4 amplitudini şi 4 diferenţe de fază, în total 16 combinaţii. Această schemă de modulare poate fi folosită la transmiterea a 4 biţi pe simbol şi este numită **QAM** (**Quadrature Amplitude Modulation**, rom: modulaţia cuadratică în amplitudine) atunci când este folosită pentru transmisia a 9600 biţi pe secundă pe o linie de 2400 baud.

Fig. 2-25(c) reprezintă încă o schemă de modulare, care implică amplitudine şi faza. Ea permite 64 de combinaţii, astfel încât pot fi transmişi 6 biţi pe simbol. Se numeşte **QAM-64**. Sunt folosite şi **QAM**-uri de ordine mai înalte.

Diagramele de genul celor din fig. 2-25, care reprezintă combinaţiile posibile de amplitudine şi fază, sunt numite **tipare de constelaţii**. Fiecare standard de modem de viteze înalte are propriul lui tipar de constelaţie şi poate comunica numai cu alte modemuri care folosesc acelaşi standard (cu toate că majoritatea modemurilor pot simula modemuri mai lente).

90 90 90

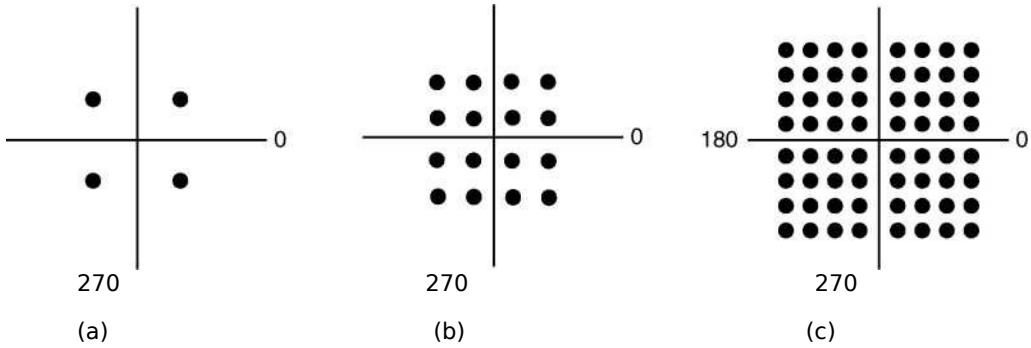
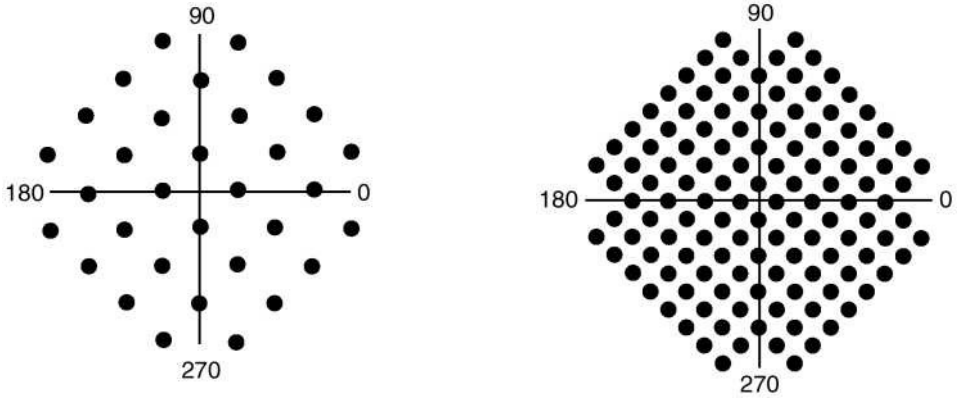


Fig. 2-25. (a) QPSK. (b) QAM-16. (c) QAM-64

Cu atâtea puncte în tiparul de constelaţie, chiar şi un mic nivel de zgomot detectat în amplitudine sau fază poate conduce la o eroare, adică la mai mulţi biţi eronaţi. Pentru a reduce posibilitatea de a genera o eroare, standardele pentru modemuri cu viteze mari fac corecţia erorilor, adăugând biţi suplimentari la fiecare eşantion. Astfel de scheme sunt cunoscute sub numele de **TCM** (**Trellis Coding Modulation**, rom: modulatie prin codificare matricială). De exemplu, modemul standard V.32 foloseşte 32 de puncte în constelaţie pentru a transmite 4 biţi de date şi un bit de paritate pe simbol, la 2400 baud, şi obţine 9600 bps cu corecţie de erori. Tiparul său de constelaţie este cel din fig. 2-26(a). Decizia de a fi „rotită" cu 45 de grade în jurul originii a fost luată din motive inginereşti; constelaţiile rotite sau nerotite au aceeaşi capacitate de informaţie.



(b) (c)

Fig. 2-26. (a) V.32 pentru 9600 Kbps. (b) V.32 bis pentru 14.400 Kbps

Următorul pas peste 9600 bps este 14.400 bps. Este numit **V.32 bis**. Această viteză este atinsă prin transmiterea a 6 biţi de date şi un bit de paritate pe eşantion la o rată de 2400 baud. Tiparul de constelaţie are 128 de puncte atunci când se foloseşte **QAM-128**, aşa cum se vede în fig. 2-26(b). Fax- modemurile folosesc această viteză pentru a transmite pagini care au fost scanate ca o hartă de biţi (bitmap). QAM-256 nu este folosit în nici unul din modemurile telefonice standard, dar este folosit în reţelele de cablu, aşa cum vom vedea mai târziu.

Următorul modem telefonic după V.32 bis este **V.34**, care atinge 28.800 bps la 2400 baud cu 12 biţi de date/simbol. Ultimul modem în aceasta serie este **V.34 bis**, care foloseşte 14 biţi de da- te/simbol la 2400 baud şi atinge 33.600 baud.

Pentru a creşte în continuare rata efectivă, multe modemuri comprimă datele înainte de a le transmite, şi pot obţine o rată efectivă de transmisie a datelor de peste 33.600 bps. Pe de alta parte, aproape toate modemurile testează linia înainte de a începe să transmită date de la utilizator şi, dacă observă că sunt probleme care ţin de calitatea transmisiei, reduc viteza sub cea maxima. Astfel, viteza efectivă observată de utilizator poate fi mai mică, egală sau mai mare faţă de cea oficială.

Toate modemurile moderne permit traficul din ambele direcţii în acelaşi timp (folosind frecvenţe diferite pentru direcţii diferite). O conexiune care permite traficul simultan în ambele direcţii se numeşte **full duplex**. O şosea cu două benzi este full duplex. O conexiune care permite traficul în oricare dintre sensuri, dar pe

rând se numeşte **half duplex**. O şină de cale ferata este **half duplex**. O conexiune care permite traficul într-o singura direcţie se numeşte **simplex**. O stradă cu sens unic este simplex. Un alt exemplu de conexiune simplex este o fibră optică cu un laser la un capăt şi un detector de lumină la celălalt.

Motivul pentru care modemurile standard se opresc la 33.600 este că limita Shannon pentru sistemul telefonic este de aproximativ 35 Kbps, aşa că o viteza mai mare ar încălca legile fizicii (departamentul termodinamică). Pentru a afla dacă modemurile de 56 Kbps sunt posibile teoretic, urmăriţi discuţia în continuare.

De ce este limita teoretică de 35 Kbps? Are de-a face cu lungimea medie a buclelor locale şi cu calitatea acestor linii. Cei 35 Kbps sunt determinaţi de lungimea medie a buclelor locale. În fig. 2-23, un apel pornit de la computerul din stânga şi terminat la ISP1 trece prin două bucle locale ca semnal analogic, una la sursă şi una la destinaţie. Fiecare dintre acestea adaugă semnalului zgomote. Dacă am putea scăpa de una dintre aceste bucle locale, rata maximă s-ar dubla.

ISP2 face exact acest lucru. Are o conexiune pur digitală de la cel mai apropiat oficiu final. Semnalul digital folosit în trunchiuri ajunge direct la ISP 2, eliminând codoarele/decodoarele, modemu- rile şi transmisia analogică finală. Astfel, când unul din capetele conexiunii este total digital, cum se întâmplă cu majoritatea ISP-urilor în zilele noastre, rata maxima de date poate fi până la 70 Kbps. Între doi utilizatori cu modemuri şi linii analogice, rata maximă este de 33,6 Kbps.

Motivul pentru care se folosesc modemurile de 56 Kbps are de-a face cu teorema lui Nyquist. Canalul telefonic este lat de aproximativ 4000 Hz (incluzând benzile suplimentare). Numărul maxim de eşantioane independente care se pot prelua pe secunda este astfel de 8000. Numărul de biţi per eşantion în SUA este 8, dintre care unul este folosit pentru comandă, permiţând astfel numai 56.000 bps pentru date de la utilizator. În Europa, toţi cei 8 biţi sunt disponibili pentru date de la utilizator, deci puteau fi folosite modemuri de 64.000 bps, dar pentru a se conveni asupra unui standard internaţional, s-a ales 56.000.

Acest standard de modem este numit **V.90**. Oferă un canal ascendent de 33,6 Kbps (utilizator către ISP) şi un canal descendent de 56 Kbps (ISP către utilizator), pentru că de obicei sunt mai multe date de transportat de la ISP la utilizator decât invers (de exemplu, cererea unei pagini web ocupă doar câţiva octeţi, în timp ce pagina efectivă poate fi de câţiva megabiţi). În teorie, un canal ascendent de peste 33,6 Kbps era posibil, dar deoarece multe bucle locale sunt prea zgomotoase chiar şi pentru 33,6 Kbps, s-a decis să se aloce mai multa lărgime de banda pentru canalul descendent, pentru a creşte şansele ca acesta să ajungă să funcţioneze la 56 Kbps.

Următorul pas dincolo de V.90 este **V.92**. Aceste modemuri sunt capabile să transfere cu 48 Kbps pe canalul ascendent, dacă linia telefonică suportă. De asemenea, ele determină viteza potrivită de funcţionare în aproximativ jumătate din timpul uzual de 30 de secunde cerut de vechile modemuri. În sfârşit, permit unui apel telefonic să întrerupă o sesiune Internet, dacă linia telefonică suporta serviciul de apel în aşteptare.

Linii digitale pentru abonat (xDSL)

Când industria telefonică a ajuns în sfârşit la 56 Kbps, s-a bătut singură pe spate ca pentru o treabă bine făcută. Intre timp, industria televiziunii prin cablu oferea

viteze de până la 10 Mbps pe cabluri partajate, iar companiile de servicii prin satelit plănuiau să ofere trafic ascendent spre satelit de 50 Mbps. Cum accesul la Internet devenise o parte din ce în ce mai importantă din afacerile lor, companiile telefonice (LEC-urile) au început să înţeleagă că aveau nevoie de un produs mai competitiv. Răspunsul găsit a fost să înceapă să ofere noi servicii digitale peste bucla locală. Serviciile cu mai multa lăţime de bandă decât serviciul telefonic standard sunt numite uneori **broadband** (rom: de bandă larga), deşi termenul este mai mult un concept de marketing decât un real concept tehnic.

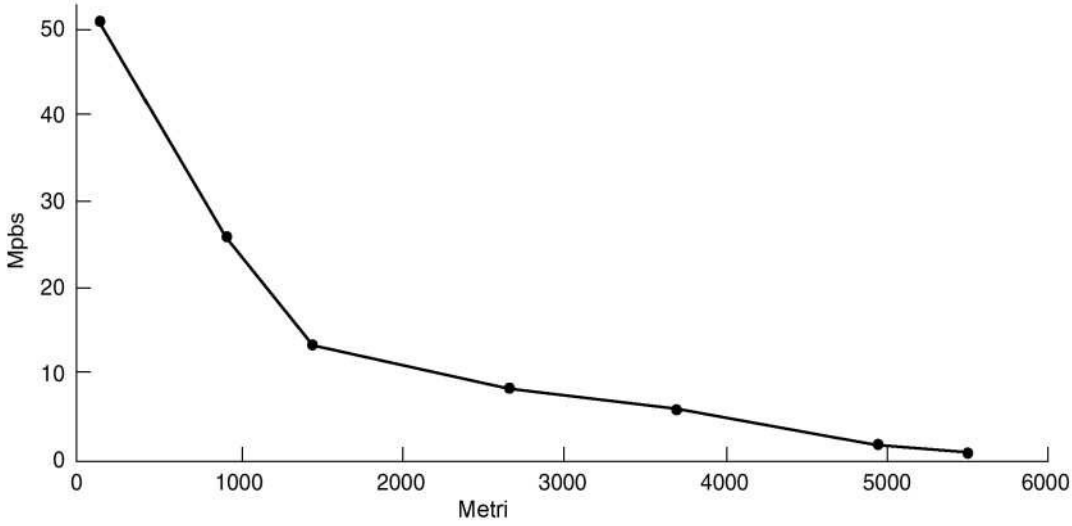


Fig. 2-27. Lăţime în funcţie de distanţă pentru UTP categoria 3 pentru DSL.

Iniţial, erau mai multe oferte care se suprapuneau, toate purtând numele generic de **xDSL** (**Digital Subscriber Line**, rom: linie digitală pentru abonat), cu diferite variabile *x*. În continuare vom discuta aceste servicii, dar ne vom concentra în primul rând spre ceea ce probabil va deveni cel mai popular dintre aceste servicii, **ADSL** (**Asymmetric DSL**, rom: DSL asimetrice). Cum ADSL încă se mai dezvolta şi nu toate standardele sunt cu totul finalizate, unele dintre detaliile date mai jos s-ar putea să se schimbe în timp. Dar ideea de baza ar trebui să rămână valabilă. Pentru mai multe informaţii despre ADSL, se poate vedea (Summers, 1999; şi Vetter et al., 2000).

Motivul pentru care modemurile sunt încete este acela că telefoanele au fost inventate pentru a transporta vocea umană şi întregul sistem a fost optimizat pentru acest scop. Datele au fost mereu copiii vitregi. În punctul în care fiecare buclă locală se conectează în oficiul final, firul intra într-un filtru care atenuează toate frecvenţele de sub 300 Hz şi peste 3400 Hz. Filtrarea nu este exactă – 300 Hz şi 3400 Hz sunt punctele de la 3 dB – aşa că lărgimea de bandă este considerată de 4000 Hz, chiar dacă distanţa dintre punctele de 3 dB este de 3100 Hz. Deci şi datele sunt restricţionate în aceasta bandă îngustă.

Trucul care face ca xDSL să funcţioneze este că linia unui client abonat la un astfel de serviciu este conectată la un tip diferit de comutator, care nu are acest filtru, făcând astfel disponibilă întreaga capacitate a buclei. Factorul limitator este

constitut de legile fizice aplicate buclei locale şi nu de lăţimea de bandă artificială de 3100 Hz, creată de filtru.

Din păcate, capacitatea buclei locale depinde de câţiva factori, inclusiv lungimea, grosimea şi calitatea la modul general. Un grafic al lăţimii de bandă potenţiale în funcţie de distanţă este prezentat în fig. 2-27. Aceasta figură presupune că toţi ceilalţi factori sunt optimi (fire noi, legături bune, etc.).

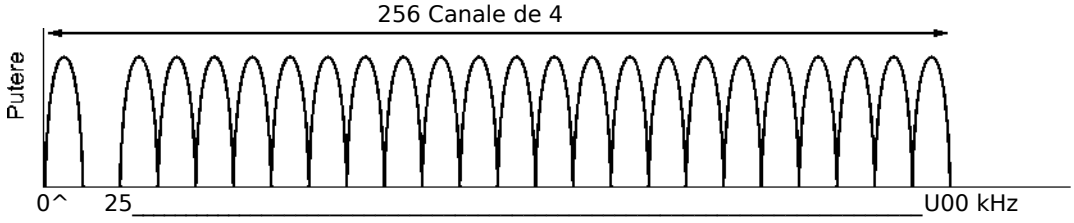
Implicaţiile acestei figuri creează o problemă pentru compania de telefoane. Când alege viteza pe care o va oferi abonaţilor, alege în acelaşi timp o rază de acţiune în funcţie de oficiile sale finale, dincolo de care serviciul nu poate fi oferit. Aceasta înseamnă că atunci când utilizatori aflaţi la distanţă încearcă să se înscrie la acest serviciu, s-ar putea să li se spună „Mulţumim mult pentru interesul dumneavoastră, dar locuiţi cu 100 de m prea departe de cel mai apropiat oficiu final prin care puteţi a beneficia de acest serviciu. Puteţi să vă mutaţi?”. Cu cât este mai mică viteza aleasă, cu atât

mai mare este raza, fiind astfel acoperiţi mai mulţi clienţi. Dar cu cât este mai mică viteza, cu atât este mai puţin atractiv serviciul şi oamenii care vor dori să plătească pentru el vor fi mai puţini. Aici afacerile se întâlnesc cu tehnologia. (O potenţială soluţie este să se construiască mini-oficii finale în toate cartierele, dar aceasta este o propunere cam scumpă.)

Serviciile xDSL au fost proiectate cu anumite scopuri. În primul rând, serviciile trebuie să funcţioneze peste buclele locale de cabluri cu perechi torsadate de categoria 3 existente. În al doilea rând, nu trebuie să afecteze telefoanele şi faxurile clienţilor. În al treilea rând, trebuie să fie mult mai rapide decât 56 Kbps. În al patrulea rând, ar trebui să funcţioneze tot timpul, contra unei taxe lunare, dar nu a unei taxe pe minut.

Oferta ADSL iniţială a venit de la AT&T şi funcţiona prin divizarea spectrului disponibil în bucla locală, care este de aproximativ 1.1 MHz, în trei benzi de frecvenţa: **POTS** (**Plain Old Telephone Service**, rom: serviciul telefonic tradiţional), canalul ascendent (de la utilizator la oficiul final) şi canalul descendent (de la oficiul final la utilizator). Tehnica de a avea mai multe benzi de frecvenţă se numeşte multiplexare prin divizarea frecvenţei; o vom studia în detaliu într-un paragraf ulterior. Ofertele care au urmat, de la alţi furnizori, au urmat o alta abordare şi se pare că aceasta va avea câştig de cauză, aşa că o vom descrie în continuare.

Abordarea alternativă, numita **DMT** (**Discrete MultiTone**, rom: ton multiplu discret), este ilustrată în figura2-28. De fapt, spectrul disponibil de 1.1 MHz al buclei locale se divizează în 256 canale independente de 4312,5 Hz fiecare. Canalul 0 este folosit pentru POTS. Canalele l-5 sunt nefolosite, pentru a preveni interferenţele între semnalele de voce şi date. Dintre cele 250 de canale rămase, unul este folosit pentru controlul fluxului ascendent şi unul pentru controlul fluxului



descendent. Restul sunt disponibile pentru date de la utilizator.

Voce Flux ascendent Flux descendent

Fig. 2-28. Funcţionarea ADSL folosind modulaţie cu ton multiplu discret

În principiu, fiecare dintre canalele rămase poate fi folosit pentru flux de date full-duplex, dar armonicele, interferenţele şi alte efecte ţin sistemele practice departe de limitele teoretice. Este la latitudinea furnizorului de servicii să determine câte canale să fie folosite pentru fluxul ascendent şi câte pentru cel descendent. Un raport de egalitate între fluxurile ascendent şi descendent este tehnic posibil, dar majoritatea furnizorilor de servicii alocă în jur de 80%-90% din lăţimea de bandă canalului descendent, deoarece majoritatea utilizatorilor primesc mai multe date

decât trimit. Această alegere dă naştere „A”-ului din ADSL. O variantă de divizare des întâlnită este alocarea a 32 de canale pentru fluxul ascendent, restul fiind alocate pentru cel descendent. De asemenea, este posibil să se folosească unele dintre canalele cu frecvenţa cea mai ridicată din fluxul ascendent în mod bidirecţional, pentru creşterea lăţimii de bandă, deşi această optimizare cere adăugarea unui circuit special pentru anularea ecourilor.

Standardul ADSL (ANSI T1.413 şi ITU G.992.1) permite viteze de până la 8 Mbps pentru fluxul descendent şi 1 Mbps pentru cel ascendent. Totuşi, puţini furnizori de servicii oferă acest flux. De obicei, furnizorii oferă 512 Kbps pentru fluxul descendent şi 64 Kbps pentru cel ascendent în cazul serviciului standard, respectiv 1 Mbps pentru flux descendent şi 256 Kbps pentru flux ascendent în cazul serviciului premium.

În cadrul fiecărui canal este folosită o schemă de modulare similară cu V.34, deşi rata de eşantionare este de 4000 baud în loc de 2400 baud. Calitatea liniilor din fiecare canal este constant monitorizata şi rata de transmisie este constant ajustată, aşa că pe canale diferite pot fi folosite rate diferite. Datele efective sunt trimise cu modulaţie QAM, cu până la 15 biţi per baud, folosind o diagramă constelaţie analoagă cu aceea din figura2-25(b). Având, de exemplu, 224 de canale de flux descendent şi 15 biţi pe baud la 4000 baud, lăţimea de bandă pe flux descendent este de 13,44 Mbps. În practică, raportul semnal-zgomot nu este destul de bun pentru a se atinge aceasta rată, dar se poate atinge rata de 8 Mbps pe distanţe scurte şi pe bucle de calitate superioara, motiv pentru care standardul merge până la aceasta valoare.

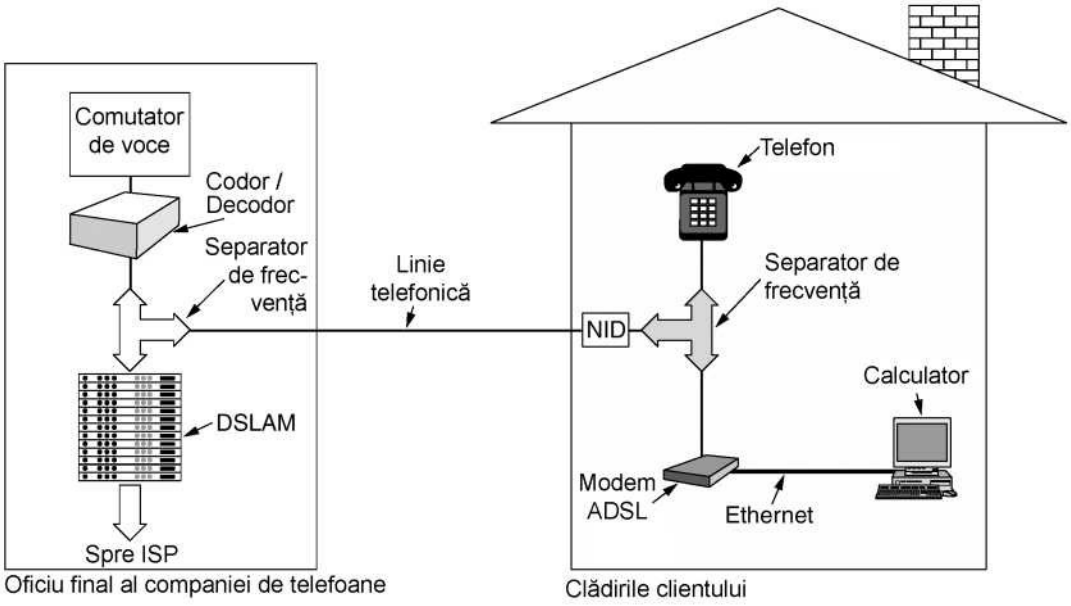


Fig. 2-29. O configuraţie tipică pentru echipamente ADSL.

O schema tipică ADSL este ilustrata în fig. 2-29. În aceasta schemă, un tehnician al companiei de telefoane trebuie să instaleze un **NID** (**Network Interface Device**,

rom: dispozitiv de interfaţare reţea) la cererea utilizatorului. Această cutie mică de plastic marchează sfârşitul proprietăţii companiei de telefoane şi începutul proprietăţii clientului. Aproape de NID (sau uneori combinat cu acesta) se află un **separator** (eng. splitter), un filtru analogic care separă din fluxul de date banda de 0-4000 Hz, folosită de POTS. Semnalul POTS este dirijat la telefonul sau faxul existent, iar semnalul de date este dirijat către modemul ADSL. Modemul ADSL este de fapt un procesor de semnal digital configurat să funcţioneze ca 250 de modemuri QAM care operează în paralel la diferite frecvenţe. Cum majoritatea modemurilor ADSL sunt externe, calculatorul trebuie sa fie conectat cu ele la o viteza mare. De obicei, aceasta se face punând o placă Ethernet în calculator şi operând o reţea Ethernet foarte mică, de două noduri, conţinând doar calculatorul şi modemul ADSL. Uneori este folosit portul USB în loc de Ethernet. Pe viitor vor fi disponibile, fără îndoială, plăci interne de modem ADSL.

La celălalt capăt al firului, pe partea oficiului final, este instalat un separator de frecvenţe similar. Aici este filtrată partea de voce a semnalului şi este trimisă către un comutator de voce normal.

Semnalul de peste 26 KHz este rutat către un nou tip de dispozitiv numit **DSLAM** (**Digital Subscriber Line Access Multiplexer**, rom: multiplexor pentru acces la linie digitală pentru abonat), care conţine acelaşi tip de procesor de semnal digital ca şi modemul ADSL. Odată ce semnalul digital recepţionat a fost convertit într-un şir de biţi, sunt formate pachete şi acestea sunt trimise ISP-ului.

Această separare completă între sistemul de voce şi ADSL simplifică furnizarea serviciilor ADSL de către companiile telefonice. Nu este nevoie decât de cumpărarea unui DSLAM şi a unui separator, urmată de ataşarea abonaţilor ADSL la separator. Alte servicii de lăţime mare de bandă (de exemplu ISDN) necesită schimbări mult mai mari la nivelul echipamentelor de comutare deja existente.

Un dezavantaj al soluţiei din figura2-29 este prezenţa NID-ului şi a separatorului în locuinţa clientului. Instalarea acestora se poate face doar de către un tehnician al companiei de telefonie, fiind nevoie de o intervenţie specială scumpă (adică de trimiterea unui tehnician la locuinţa clientului). Prin urmare, s-a standardizat o variantă alternativă fără separator. Neoficial este numit G.lite dar numărul de standard ITU este G.992.2. Este acelaşi ca în fig. 2-29, dar fără separator. Linia telefonică existentă este folosită fără nici o modificare. Singura diferenţă este că microfiltrul trebuie să fie introdus în fiecare mufă de telefon dintre telefon sau modem ADSL şi fir. Microfiltrul pentru telefon este un filtru-trece-jos care elimină frecvenţele mai mari de 3400 Hz; microfiltrul pentru ADSL este un filtru-trece-sus care elimină frecvenţele sub 26 KHz. Oricum, acest sistem nu este la fel de fiabil ca şi cel cu separator, deci G.lite poate fi folosit doar până la 1,5 Mbps (faţă de cei 8 Mbps de la ADSL cu separator). G.lite are nevoie oricum de separator la oficiul terminal, dar instalarea nu mai necesită mii de drumuri pentru intervenţii la clienţi.

ADSL e doar un standard pentru nivelul fizic. Ceea ce rulează la nivelurile superioare depinde de distribuitorul de Internet. Deseori, acesta alege ATM datorită posibilităţii ATM-ului de a satisface calitatea serviciului şi datorită faptului că multe companii telefonice conţin ATM la baza reţelei.

Bucle locale fără fir

Din 1996 în SUA şi ceva mai târziu în alte ţări, companiile care doreau să intre în competiţie cu puternicele companii locale de telefonie (fostele monopoluri), numite **ILEC** (**Incumbent LEC**, rom: LEC-uri de facto), sunt libere să o facă. Cei mai probabili candidaţi sunt companiile de telefoane pe distanţă lungă (IXC-urile). Orice IXC care dorea să intre în telefonia locală trebuia să îndeplinească unele condiţii. În primul rând, trebuie să cumpere sau să închirieze o clădire pentru primul oficiu terminal dintr-un oraş. În al doilea rând, trebuie să pună comutatoare de telefoane şi alte echipamente în oficiul terminal, dispozitive care sunt puse în vânzare de diverşi producători. În al treilea rând, trebuie tras un cablu cu fibră optică între oficiul terminal şi cel mai apropiat oficiu, pentru ca noii consumatori să aibă acces la reţeaua naţională. În al patrulea rând, trebuie să racoleze clienţi, de obicei prin reclamă care anunţă preţuri mai mici şi servicii mai bune decât cele ale ILEC.

De aici începea parte grea. Să presupunem că apar câţiva clienţi. Cum are de gând noua companie de telefoane locală, numită **CLEC** (**Competitive LEC**, rom: LEC competitivă) să conecteze telefoanele clienţilor la oficiul final proaspăt deschis? Obţinerea drepturilor necesare şi întinderea firelor sau a fibrei sunt acţiuni foarte costisitoare. Mute CLEC-uri au descoperit o alternativă la bucla tradiţională de cablu torsadat: **WLL-ul (Wireless Local Loop**, rom: bucla locală fără fir).

Într-un anumit fel, un telefon fix care foloseşte o buclă locală fără fir seamănă cu un telefon mobil, dar sunt trei diferenţe importante. Prima: clienţii din bucla locală fără fir doresc deseori Internet de mare viteză, la viteze care să egaleze ADSL-ul. A doua: noul client nu are probabil nimic împotrivă ca un tehnician al CLEC să instaleze o antenă mare pe acoperişul său, direcţionată către oficiul CLEC. A treia: utilizatorul nu se mută, eliminând toate problemele de mobilitate şi timpii morţi datoraţi celulelor despre care vom vorbi mai târziu în acest capitol. Şi astfel se naşte o nouă industrie: **fixă fără fir (fixed wireless)** (telefonie locală şi servicii Internet oferite de CLEC pe o buclă locală fără fir).

Deşi WLL şi-a început activitatea semnificativă în 1998, trebuie să ne întoarcem în 1969 pentru a-i vedea originile. În acel an, FCC a alocat două canale TV (de 6MHz fiecare) pentru televiziunea educativă la 2,1GHz. În anii ce au urmat, au mai fost adăugate 31 de canale la 2,5GHz, cu un total de 198 MHz.

Televiziunea educativă nu a prins, iar în 1998 FCC a retras frecvenţele şi le-a alocat radioului bidirecţional. Au fost imediat acaparate de buclele locale fără fir. La aceste frecvenţe, microundele au 10-12 cm. Au un domeniu de 50 km şi penetrează vegetaţia şi ploaia destul de bine. Cei 198 MHz noi din spectru au fot puşi în uz pentru buclele locale fără fir ca un serviciu numit **MMDS** (**Multichannel Multipoint Distribution Service**, rom: serviciu de distribuţie multicanal multipunct). MMDS poate fi privit ca un MAN (Reţea de acoperire Metropolitană), la fel ca şi varul său LMDS (discutat mai jos).

Marele avantaj al acestui serviciu este că tehnologia este bine stabilită şi echipamentele sunt disponibile. Dezavantajul este că lăţimea de bandă disponibilă este modestă şi trebuie folosită în comun de mulţi utilizatori dintr-o arie geografică desul de mare.

Lăţimea mică de bandă a MMDS a făcut din undele milimetrice o alternativă interesantă. La 28-31 GHz în SUA şi 40 GHz în Europa nu se alocau frecvenţe deoarece este dificil să construieşti circuite integrate cu siliciu atât de rapide.

Problema a fost rezolvată de inventarea circuitelor integrate cu galiu şi arseniu, deschizându-se astfel banda milimetrică pentru radio-comunicaţii. FCC a răspuns cererii alocând 1,3 GHz unei noi bucle locale fără fir numită **LMDS** (**Local Multipoint Distribution Service**, rom: serviciu local de distribuţie multipunct). Această alocare este cea mai mare alocare de lăţime de bandă pe care a făcut-o FCC vreodată. O lăţime similară este alocată şi în Europa, dar la 40 GHz.

Modul de operare al LMDS este prezentat în fig. 2-30. În figură este prezentat un turn cu mai multe antene, fiecare fiind îndreptată într-o altă direcţie. Cum razele milimetrice sunt foarte bine di- recţionate, fiecare antenă defineşte un sector, independent de celelalte. La această frecvenţă, raza de acţiune este de 2-5 km, ceea ce înseamnă că e nevoie de multe turnuri pentru a acoperi un oraş întreg.

Ca şi ADSL, LMDS foloseşte o alocare de lăţime de bandă asimetrică, favorizând canalul de recepţie. Cu tehnologia curentă, fiecare sector poate avea 36 Gbps pentru recepţie şi 1 Mbps pentru transmisie, bandă folosită în comun de toţi utilizatorii sectorului. Dacă fiecare utilizator activ descarcă trei pagini de 5 KB pe minut, utilizatorul ocupa în medie 200bps din spectru, ceea ce permite maxim 18000 de utilizatori activi pe sector. Totuşi, pentru a se menţine întârzierile la un nivel rezonabil, ar trebui să nu fie mai mult de 9.000 de utilizatori. Cu patru sectoare, ca în fig. 2-30, poate fi deservită o populaţie activă de 36.000 de locuitori. Presupunând ca unul din trei utilizatori este conectat în momentele de vârf, un singur turn cu patru antene poate deservi 100.000 de oameni într-o rază de 5 km faţă de turn. Aceste calcule au fost făcute de multe CLEC-uri, dintre care unele au ajuns la concluzia că făcând o investiţie modestă în turnuri pentru unde milimetrice pot oferi utilizatorilor viteze comparabile cu cablul TV la un preţ mai mic.

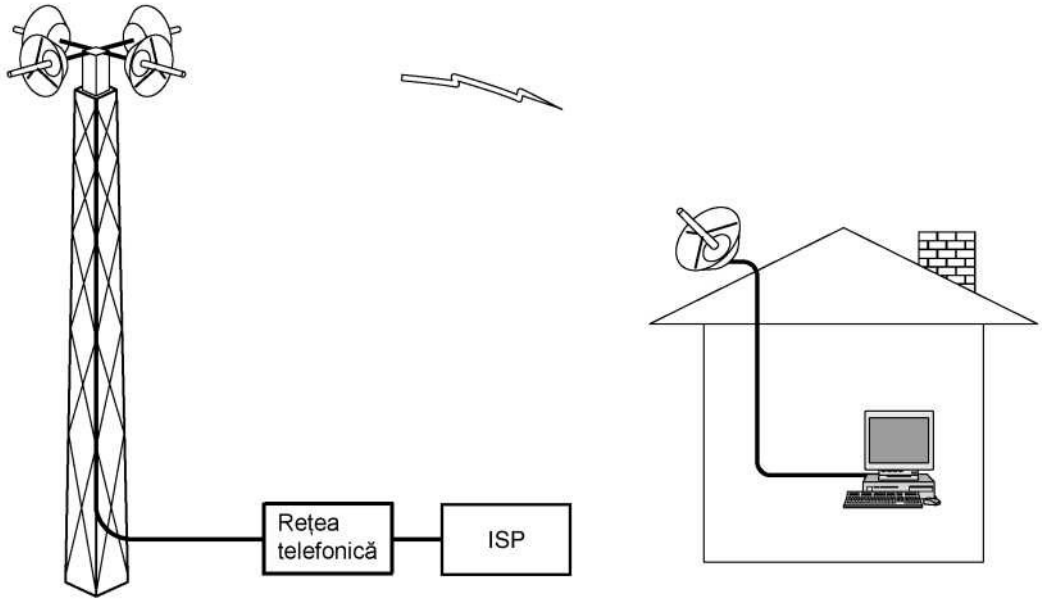


Fig. 2-30. Arhitectura unui sistem LMDS.

LMDS are totuşi câteva probleme. Pentru început, undele milimetrice se propagă în linie dreaptă, deci trebuie să existe vizibilitate între antena de pe acoperiş şi turn.

O alta este că frunzele absorb aceste unde destul de bine, deci turnul trebuie să fie suficient de înalt pentru ca drumul până la antenă să treacă pe deasupra copacilor. Iar ceea ce pare un drum liber în decembrie s-ar putea să nu mai fie în iulie, când copacii sunt plini de frunze. Ploaia absoarbe şi ea aceste unde. Totuşi, erorile introduse de ploaie pot fi compensate cu un cod corector de erori sau prin mărirea puterii atunci când plouă. Oricum, serviciul LMDS e mult mai probabil să fie introdus mai întâi în ţinuturile uscate, să zicem în Arizona, decât în Seattle.

Buclele locale fără fir nu au şanse să prindă dacă nu există standarde, pentru a încuraja vânzătorii de echipamente să producă şi să asigure clienţii că pot schimba CLEC-ul fără să fie nevoie să cumpere un alt echipament. Pentru a asigura acest standard, IEEE a întrunit un comitet numit 802.16 pentru a stabili standardul pentru LMDS. Standardul 802.16 a fost publicat în aprilie 2002. IEEE numeşte 802.16 **MAN fără fir** (**wireless MAN**). IEEE 802.16 a fost proiectat pentru telefonia digitală, acces la Internet, conectarea a două LAN-uri îndepărtate, emisie radio şi TV, precum şi pentru alte utilizări. Îl vom studia mai în detaliu în cap. 4.

2.5.4 Trunchiuri şi multiplexare

Economiile rezultate din scalabilitate joacă un rol important în sistemul telefonic. Instalarea şi întreţinerea unor trunchiuri de bandă largă între două oficii de comutare costă cam tot atât cât instalarea şi întreţinerea unui trunchi de bandă joasă (costul provine de la săparea şanţului, nu de la firul de cupru sau de la fibra optică). În consecinţă, companiile telefonice au dezvoltat metode sofisticate pentru multiplexarea mai multor convorbiri pe aceeaşi magistrală fizică. Aceste metode de multiplexare se pot împărţi în două categorii principale: **FDM** (**Frequency Division Multiplexing**, rom: multiplexare cu divizare în frecvenţă) şi **TDM** (**Time Division Multiplexing**, rom: multiplexare cu divizare în timp). La FDM, spectrul de frecvenţă este împărţit în mai multe canale logice, fiecare utilizator având drepturi exclusive asupra unei anumite benzi de frecvenţă. La TDM, utilizatorii îşi aşteaptă rândul (în mod repetat, circular), fiecare utilizator obţinând întreaga bandă de frecvenţă pentru o scurtă perioadă.

Difuzarea radio AM ilustrează ambele metode de multiplexare. Spectrul alocat este de aproape 1MHz, aproximativ între 500 şi 1500 de KHz. Pentru diferite canale logice (staţii) sunt alocate frecvenţe diferite. Fiecare canal logic operează într-un anumit domeniu al spectrului, distanţele între canale fiind destul de mari pentru a preveni interferenţa. Acest sistem este un exemplu de multiplexare prin divizarea frecvenţei. În plus (în unele ţări), staţiile individuale au două subcanale logice: muzică şi publicitate. Acestea două alternează în timp pe aceeaşi frecvenţă, la început muzică şi după un timp o secvenţă de reclame, apoi din nou muzică şi aşa mai departe. Această situaţie se numeşte multiplexare prin divizare în timp.

În continuare, vom studia multiplexarea prin divizarea în frecvenţă. După aceea vom analiza cum se poate aplica FDM fibrelor optice (multiplexare prin divizarea lungimii de undă). Apoi ne vom întoarce la TDM, iar în final vom studia un sistem TDM avansat folosit în fibrele optice (SONET).

Multiplexarea prin divizarea în frecvenţă

Fig. 2-31 ne prezintă cum sunt multiplexate trei canale de bandă vocală folosind

FDM. Filtrele limitează lărgimea de bandă folosită la 3100 de Hz pe canal de bandă vocală. Atunci când sunt multi- plexate împreună mai multe canale, fiecărui canal îi sunt alocaţi 4000 de Hz, astfel încât canalele să fie bine separate. Mai întâi, canalele de voce sunt deplasate în frecvenţă, fiecare cu o valoare diferită. Apoi ele pot fi combinate, deoarece nu există două canale care să ocupe aceeaşi zonă a spectrului. Este de remarcat că, deşi există spaţii (spaţii de gardă) între canale, poate să apară o suprapunere între canalele adiacente, deoarece filtrele nu au marginile abrupte. Această suprapunere înseamnă că un semnal puternic la capătul unui canal va fi simţit în canalul adiacent ca un zgomot non-termic.

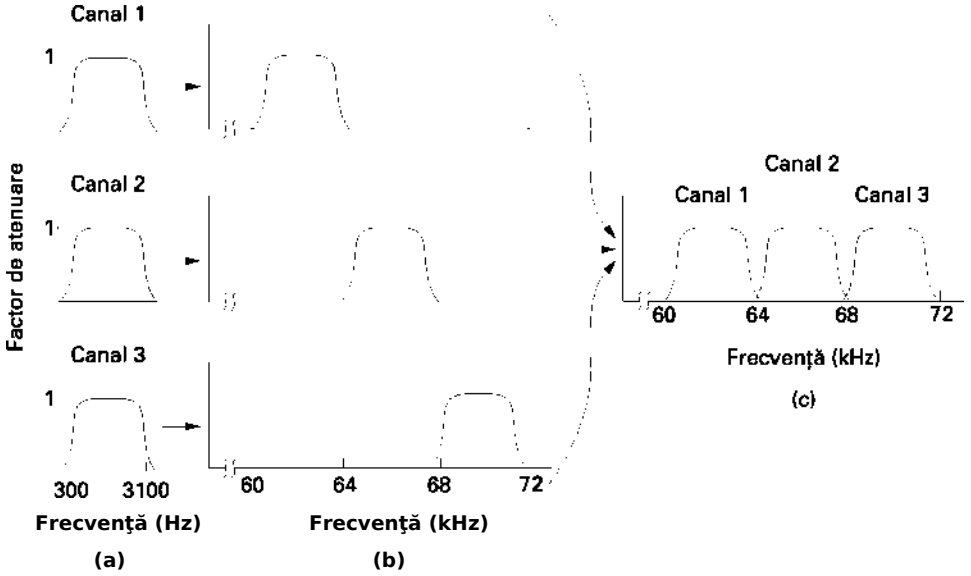


Fig. 2-31. Mutliplexarea prin divizare în frecvenţă. (a) Banda de frecvenţă iniţială. (b) Banda deplasată în frecvenţă. (c) Canalul multiplexat.

Schemele FDM folosite pe glob sunt, până la un anumit nivel, standardizate. Un standard foarte folosit este dat de 12 canale vocale a 4000 Hz multiplexate în banda de 60 până la 108 KHz. Această unitate este numită **grup**. Banda 12 – 60 KHz este uneori folosită de un alt grup. Multe companii oferă clienţilor un serviciu de linie închiriată de 48 până la 56 Kbps, bazat pe un grup. Cinci grupuri (60 de canale vocale) formează un **super-grup**. Următoarea unitate este un **master-grup**, care este format din cinci super-grupuri (în standardul CCITT) sau zece super-grupuri (sistemul Bell). Există şi alte standarde, care cuprind până la 230.000 canale vocale.

Multiplexarea prin divizarea lungimii de undă

Pentru canalele de fibră optică se utilizează o alternativă a multiplexării prin divizarea în frecvenţă. Aceasta se numeşte **WDM** (**Wavelength Division Multiplexing**, rom: multiplexarea prin divizarea lungimii de undă). Principiul de bază pentru WDM pe fibre este prezentat în fig. 2-32. Aici se întâlnesc patru fibre la nivelul unui combinator optic, fiecare cu energia şi lungimea de undă proprie. Cele patru raze sunt combinate într-o singură fibră comună pentru a fi transmise către o destinaţie depărtată. La destinaţie, raza este despărţită în atâtea fibre câte au fost

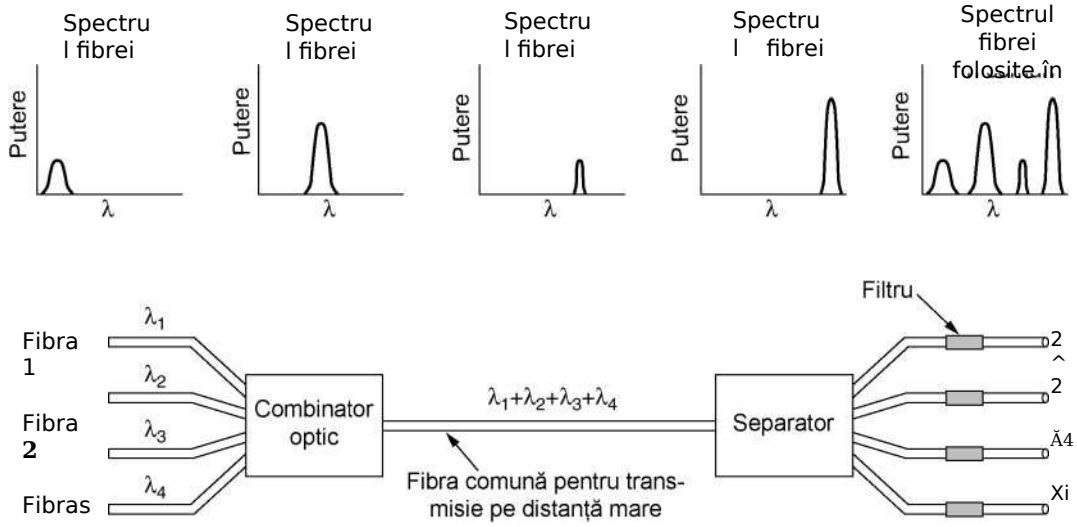


Fig. 2-32. Multiplexarea prin divizarea

iniţial. Fiecare fibră de la destinaţie conţine un mic filtru special construit, care filtrează toate lungimile de undă mai puţin una. Semnalele rezultate pot fi rutate către destinaţie sau recombinate în diferite feluri pentru

Aici nu este de fapt nimic nou. Este tot multiplexare prin divizarea în frecvenţă, aplicată la frecvenţe foarte mari. Atâta timp cât fiecare canal are propriul domeniu de frecvenţă (lungime de undă) şi toate aceste domenii sunt disjuncte, ele pot fi multiplexate împreună pe o fibră de distanţă mare. Singura diferenţă faţă de FDM prezentat anterior este aceea că sistemul optic, folosind o reţea de difracţie, este total pasiv şi, de aceea, foarte sigur.

Tehnologia WDM a progresat extrem de rapid, umbrind chiar şi tehnologia

calculatoarelor. WDM a fost inventată în jurul anului 1990. Primul sistem comercial avea opt canale, fiecare de 2,5 Gbps. În 1998 existau deja pe piaţă sisteme cu 40 de canale de 2,5 Gbps fiecare. În 2001, existau pro- produse cu 96 de canale de 10 Gbps fiecare, deci un total de 960 Gbps. Această lăţime de bandă este suficientă pentru a transmite filme întregi într-o secundă (format MPEG-2). În laboratoare se lucrează deja la sisteme cu 200 de canale. Atunci când numărul de canale este foarte mare şi lungimile de undă sunt foarte apropiate, de exemplu la 0,1 nm, sistemul mai este numit şi **DWDM (Dense WDM**, rom: WDM densă).

Este de remarcat că WDM este foarte populară datorită faptului că energia pe o singură fibră este, de obicei, cuprinsă într-o bandă de numai câţiva gigaherţi, deoarece în acest moment este imposibilă conversia mai rapidă de la mediul electric la cel optic. Prin folosirea în paralel a mai multor canale, de lungimi de undă diferite, lăţimea de bandă creşte liniar cu numărul de canale. Din moment ce banda de frecvenţă a unei singure fibre este de aproximativ 25.000 GHz (vezi fig. 2-6), există posibilitatea de a se ajunge la 2500 de canale de 10Gbps fiecare, chiar şi la o rată de 1 bit/Hz (fiind posibile de asemenea şi rate mai mari).

O altă direcţie de dezvoltare o reprezintă toate amplificatoarele optice. Înainte vreme, la fiecare 100km era obligatorie separarea tuturor canalelor şi conversia fiecărui semnal optic într-un semnal electric; fiecare semnal era amplificat separat înainte de a se face conversia inversă, din semnal electric în semnal optic, urmată de recombinarea semnalelor optice. În prezent, toate amplificatoarele optice pot regenera întregul semnal odată la 1000km fără a fi nevoie de conversii multiple între semnalele electrice şi optice.

În exemplul din fig. 2-32, avem un sistem cu lungimi de undă fixe. Unii biţi din fibra de intrare 1 trec în fibra de ieşire 3, biţi din fibra de intrare 2 trec în fibra de ieşire 1 etc. Oricum, este posibil să construim şi sisteme WDM comutate. În cazul unui astfel de dispozitiv, filtrele de ieşire sunt reglabile folosind interferometre Fabry-Perot sau Mach-Zender. Pentru informaţii suplimentare despre WDM şi aplicaţiile sale în comutarea pachetelor în Internet vezi (Elmirghani şi Mouftah, 2000; Hunter şi Andonovic, 2000; Listam şi alţii, 2001)

Multiplexarea prin divizarea în timp

Tehnologia WDM este minunată, dar există încă multe fire din cupru în reţeaua telefonică, deci să revenim un timp la discuţiile referitoare la acestea. Cu toate că FDM este folosită încă pe firele de cupru sau pe canalele de microunde, ea necesită circuite analogice şi nu poate fi realizată de un calculator. Dimpotrivă, TDM-ul poate fi în întregime tratat de electronica digitală, devenind mult mai răspândit în ultimii ani. Din păcate, TDM-ul poate fi folosit numai pentru date digitale. Deoarece buclele locale produc semnale analogice, este necesară o conversie analogic – digital în oficiul final, unde toate buclele locale individuale se întâlnesc pentru a forma o magistrală.

Vom arunca acum o privire asupra modului în care mai multe semnale vocale analogice sunt di- gitizate şi sunt reunite pentru a forma o singură magistrală digitală. Datele trimise de un calculator prin modem sunt tot analogice, deci următoarea descriere se aplică şi pentru acestea. Semnalele analogice sunt digitizate în oficiul final de un echipament numit **codec** (codificator/decodificator),

operaţie care are ca rezultat o serie de numere de 8 biţi. Codec-ul realizează 8000 de eşantioane pe secundă (125 psec/eşantion), deoarece teorema lui Nyquist spune că aceasta rată este suficientă pentru a capta toată informaţia de pe canalul telefonic de 4 KHz. La o rată mai mică de eşantionare, o parte din informaţie ar putea fi pierdută; la o rată mai mare, nu se furnizează nici un fel de informaţie suplimentară. Această tehnică se numeşte **PCM** (**Pulse Code Modulation**, rom: modularea în cod de impulsuri). PCM constituie nucleul sistemelor telefonice moderne. Ca o consecinţă, practic toate intervalele de timp dintr-un sistem telefonic sunt multipli de 125 psec.

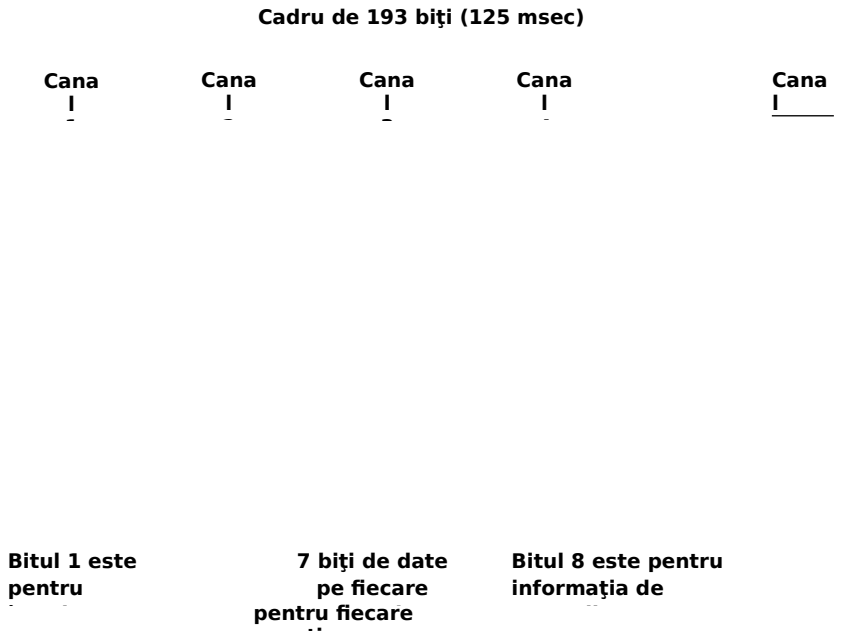


Fig. 2-33. Purtătoarea T1 (1.544 Mbps).

Când transmisia digitală a început să devină o tehnologie accesibilă, CCITT nu a fost capabil să ajungă la o soluţie cu privire la un standard pentru PCM. În consecinţă, acum există o varietate de scheme incompatibile, folosite în diverse ţări din întreaga lume.

O metodă răspândită în America de Nord şi Japonia o reprezintă purtătoarea **T1**, prezentată în fig. 2-33. (Din punct de vedere tehnic, formatul se numeşte DS1 iar purtătoarea se numeşte T1, dar noi nu vom face această deosebire subtilă aici). Purtătoarea T1 constă din 24 de canale vocale multi- plexate împreună. De obicei, cele 24 de semnale analogice sunt eşantionate pe rând, iar fluxul analogic rezultat este trecut prin codec, în loc să se utilizeze 24 de codec-uri separate a căror semnale de ieşire să compună ieşirea digitală. Fiecare canal din cele 24 va introduce 8 biţi în secvenţa de la ieşire. Şapte biţi reprezintă date, iar unul este de control, rezultând 7x8000 = 56.000 bps de date şi 1x8000 = 8.000 bps de informaţie de semnalizare pe fiecare canal.

Un cadru constă din 24x8 = 192 biţi, plus un bit suplimentar pentru încadrare, rezultând 193 de biţi la fiecare 125 ţisec. De aici rezultă o viteză de transfer a datelor de 1.544 Mbps. Bitul 193 este folosit pentru sincronizarea cadrelor, şi urmează un model de forma 0101010101... . În mod normal, receptorul verifică în continuu acest bit, pentru a fi sigur că nu a pierdut sincronizarea. Dacă într- adevăr

a pierdut tactul, receptorul poate căuta după acest şablon pentru a se resincroniza. Clienţii analogici nu pot genera un semnal conform cu acest şablon, deoarece el corespunde unei unde sinusoidale la 4000 Hz, care ar fi eliminată prin filtrare. Clienţii digitali pot, desigur, să genereze un semnal care să respecte acest şablon, dar şansele apariţiei lui sunt mici atunci când un cadru este pierdut. Atunci când un sistem T1 este folosit integral pentru date, doar 23 de canale sunt folosite pentru date. Canalul 24 este folosit pentru un şablon de sincronizare special, care permite reluarea mai rapidă a funcţionării în cazul în care un cadru este pierdut.

În cele din urmă, când CCITT a ajuns la o înţelegere, a considerat că a folosi 8000 bps de informaţie de semnalizare este o exagerare, aşa că standardul de 1.544 Mbps se bazează pe date de 8 biţi în loc de 7 biţi; aceasta înseamnă că semnalul analogic este codificat folosind 256 de niveluri discrete în loc de 128. Există două situaţii, incompatibile. În **semnalizarea prin canal comun** (**common-channel signaling**), bitul suplimentar (care este ataşat mai degrabă în urma, şi nu în faţa cadrului de 193 de biţi) are valorile 1010101010... în cadrele impare şi conţine informaţii de semnalizare pentru toate canalele în cadrele pare.

În cealaltă alternativă, **semnalizarea pe canale asociate** (**channel-asssociated signaling**), fiecare canal are propriul său subcanal de semnalizare. Un subcanal privat este constituit prin alocarea unui bit pentru semnalizare (dintre cei 8 biţi utili) la fiecare 6 cadre; astfel, 5 din 6 eşantioane sunt de 8 biţi lungime, iar ultimul este de doar 7 biţi lungime. De asemenea, CCITT a recomandat o purtătoare PCM de 2.048 Mbps numită **E1**. Această purtătoare are 32 de eşantioane de 8 biţi, împachetate în cadrul de bază de 125 psec. Treizeci dintre aceste canale sunt folosite pentru informaţii, iar celelalte două sunt folosite pentru semnalizare. Fiecare grup de 4 cadre asigură 64 de biţi de semnalizare, dintre care jumătate sunt folosiţi pentru semnalizarea pe canalul asociat şi jumătate pentru sincronizarea cadrelor sau pentru alte operaţii, în funcţie de ţările unde sunt utilizate. Purtătoarea E1 de 2.048 Mbps este larg folosită în afara Americii de Nord şi a Japoniei.

Odată ce semnalul vocal a fost codificat, este tentant să folosim metode statistice pentru a reduce numărul de biţi necesari pentru fiecare canal. Aceste tehnici sunt potrivite nu numai pentru codificarea vorbirii, ci şi pentru digitizarea oricărui semnal analogic. Toate metodele de compactare se bazează pe principiul că semnalul se modifică relativ încet în comparaţie cu frecvenţa de eşantionare, deci multe dintre informaţiile codificate pe 7 sau 8 biţi sunt redundante.

Metoda, numită **modulare diferenţială în cod de impulsuri** (**differential pulse code modulation**), constă în furnizarea la ieşire nu a amplitudinii digitizate, ci a diferenţei dintre valoarea curentă şi cea anterioară. Deoarece sunt puţin probabile salturi mai mari de ±16 pe o scală de 128, ar putea fi sufi- cienţi 5 biţi în loc de 7. Dacă semnalul are salturi prea bruşte, logica de codificare poate necesita mai multe perioade de eşantionare pentru „a prinde din urmă” aceste variaţii. În cazul vorbirii, eroarea introdusă poate fi ignorată.

O variantă a acestei metode de compactare impune ca fiecare valoare

eşantionată să difere de precedenta prin +1 sau -1. În aceste condiţii, este transmis un singur bit, care indică dacă nivelul curent este superior sau inferior nivelului precedent. Această tehnică, numită **modulare delta**, este ilustrată în fig. 2-34. La fel ca toate tehnicile de compactare care presupun schimbări mici de nivel între eşantioane consecutive, dacă sistemul se schimbă prea brusc, modularea delta poate eşua, aşa cum se arată în figură. Când se întâmplă acest lucru, informaţia se pierde.

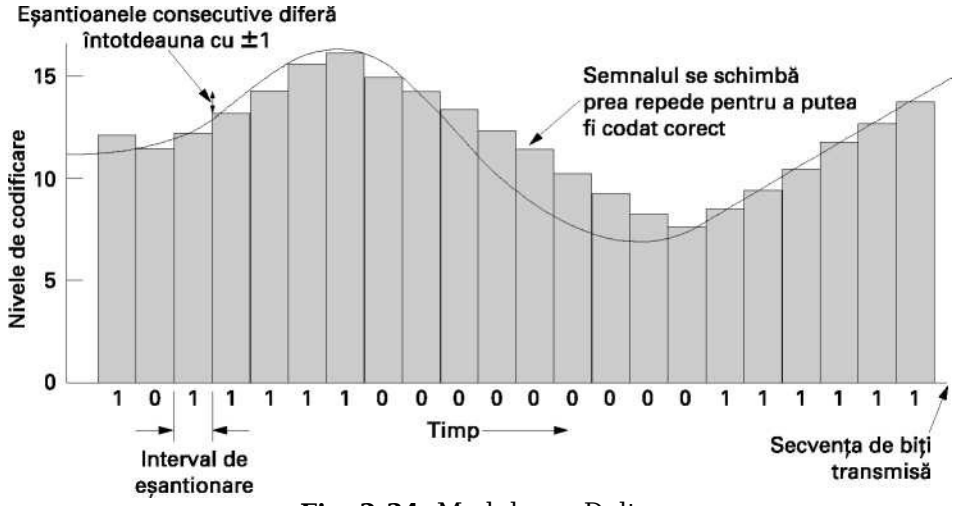


Fig. 2-34. Modularea Delta.

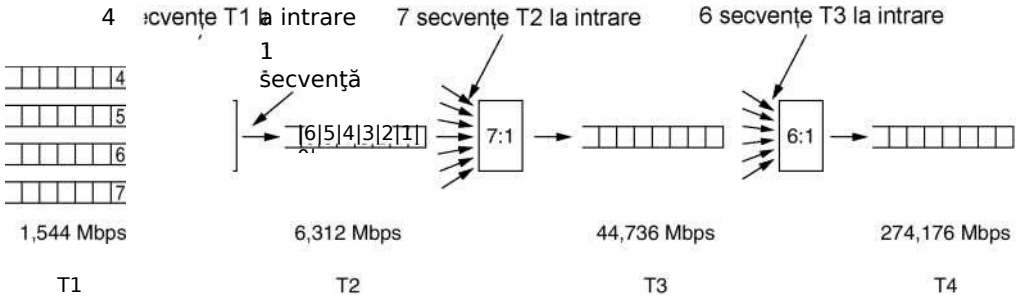


Fig. 2-35. Multiplexarea secvenţelor T1 în purtătoare cu

O îmbunătăţire pentru PCM diferenţial se obţine dacă se extrapolează câteva valori precedente pentru a prezice noua valoare şi apoi se codifică diferenţa dintre semnalul actual şi cel prevăzut. Desigur, atât transmiţătorul cât şi receptorul trebuie să folosească acelaşi algoritm de predicţie. O astfel de schemă se numeşte **codificare predictivă**. Această tehnică este utilă, pentru că reduce dimensiunea datelor care trebuie codificate şi, prin urmare, reduce dimensiunea datelor care trebuie transmise.

Multiplexarea prin divizare în timp permite ca mai multe purtătoare T1 să fie multiplexate împreună în purtătoare de un grad mai înalt. Fig. 2-35 arată cum se poate face acest lucru. În stânga se văd patru canale T1 multiplexate într-un canal T2. Multiplexarea la T2 şi peste T2 se face bit cu bit şi nu octet cu octet, ca în cazul celor 24 de canale care constituie un cadru T1. Patru secvenţe T1 la 1.544 Mbps ar trebui să genereze 6.176 Mbps, dar T2 transmite de fapt la 6.312 Mbps. Biţii suplimentari sunt folosiţi la încadrare şi la recuperare, în cazul în care purtătoarea este pierdută. T1 şi T3 sunt folosite pe scară largă de consumatori, în timp ce T2 şi T4 sunt folosite numai în cadrul sistemului de telefonie, deci nu sunt foarte cunoscute.

La nivelul următor, şapte T2 sunt combinate pentru a forma o secvenţă T3. Apoi şase T3 sunt grupate pentru a forma o secvenţă T4. La fiecare pas, se adăugă, pentru încadrare şi recuperare, o mică supraîncărcare în cazul în care sincronizarea dintre transmiţător şi receptor se pierde.

Aşa cum există neînţelegeri privind purtătoarea de bază, între Statele Unite şi restul lumii, tot aşa există neînţelegeri privitoare la modul în care se face multiplexarea în purtătoarele de bandă mai largă. Ierarhia implementată în S.U.A., care este făcută prin grupuri de 4, 7 şi 6 secvenţe, nu s-a impus ca un standard, standardul CCITT multiplexând 4 secvenţe în una singură la fiecare nivel. De asemenea, încadrarea şi recuperarea datelor se fac diferit. Ierarhia CCITT cu 32, 128, 512, 2048 şi 8192 de canale funcţionează la viteze de 2.048, 8.848, 34.304, 139.264 şi 565.148 Mbps.

SONET / SDH

La apariţia fibrelor optice, fiecare companie telefonică avea propriul sistem optic TDM. După ce AT&T-ul a fost divizat în 1984, companiile telefonice locale au fost obligate să se conecteze la diverse companii de telecomunicaţii pe distanţe mari,

fiecare companie având sisteme TDM diferite; a devenit astfel evidentă necesitatea unei standardizări. În 1985, Bellcore, divizia de cercetare a RBOC, a început să lucreze la un nou standard, numit **SONET** (**Synchronous Optical NETwork**, rom: reţea optică sincronă). Mai târziu s-a alăturat şi CCITT, fapt care s-a materializat în 1989 prin standardul SONET şi printr-un set paralel de recomandări CCITT (G.707, G.708 şi G.709). Recomandările CCITT sunt numite **SDH** (**Synchronous Digital Hierarchy**, rom: ierarhie digitală sincronă), dar diferă de SONET numai în mică măsură. Practic, aproape tot traficul pe distanţe mari din Statele Unite şi cea mai mare parte a traficului din alte zone foloseşte acum trunchiuri cu SONET pe nivelul fizic. Pentru informaţii suplimentare vezi (Bellamy 2000; Goralski, 2000; şi Shepard, 2001).

Proiectul SONET a urmărit patru obiective principale. Primul şi cel mai important, SONET trebuia să permită conlucrarea mai multor companii de telecomunicaţii. Pentru atingerea acestui obiectiv a fost necesară definirea unui standard comun de codificare a semnalului care să facă referire la lungimea de undă, la sincronizare, la structura cadrelor etc.

În al doilea rând, erau necesare câteva metode de a unifica sistemele digitale din S.U.A., Europa şi Japonia, toate bazându-se pe canale PCM de 64 Kbps, însă combinate diferit şi devenind totodată incompatibile între ele.

În al treilea rând, SONET trebuia să permită multiplexarea mai multor canale digitale. Atunci când a fost elaborat SONET, cea mai rapidă purtătoare digitală utilizată efectiv pe scară largă în Statele Unite era T3, la 44.736 Mbps. T4 era definit, dar nu era atât de folosit, iar deasupra vitezei lui T4 nu era definit nimic. O parte a misiunii SONET era să ridice ierarhia la nivel de gigabiţi/sec şi chiar mai mult. Era de asemenea necesară o modalitate standard de multiplexare a canalelor mai lente într-un canal SONET.

În al patrulea rând, SONET trebuia să asigure suportul de operare, administrare şi întreţinere (OAM – Operations, Administration, Maintenance). Sistemele precedente nu au realizat acest lucru foarte bine.

O decizie anterioară era să se facă din SONET un sistem TDM tradiţional, în care toată banda de frecvenţă a fibrei optice atribuită unui singur canal să conţină diferite intervale de timp pentru subcanale diferite. Astfel, SONET este un sistem sincron. El este controlat de un ceas principal, cu o eroare de aproximativ 10^{-9}. Biţii sunt transmişi pe o linie SONET la intervale extrem de precise, controlate de ceasul principal. Atunci când comutarea celulelor a fost propusă ca bază pentru ATM în bandă largă, faptul că ea permitea sosirea neregulată a celulelor a determinat etichetarea sa ca mod de transfer *asincron* (ATM), pentru a contrasta astfel cu operaţiile sincrone ale SONET-ului. Cu SONET, transmiţătorul şi receptorul sunt legaţi de un ceas comun; cu ATM nu sunt.

Cadrul de bază SONET este un bloc de 810 octeţi, lansat la fiecare 125 psec. Deoarece SONET este sincron, cadrele sunt emise, chiar dacă nu există date utile de transmis. Rata de 8000 cadre/sec coincide cu viteza de eşantionare a canalelor PCM folosite în sistemele telefonice digitale.

Cadrele SONET de 810 octeţi sunt cel mai bine descrise prin matrice cu 90 de coloane şi 9 rânduri. Cei 8x810 = 6480 biţi ai unui cadru sunt transmişi de 8000 de ori pe secundă, la o viteză de transfer a datelor de 51,84 Mbps. Acesta este canalul de bază SONET şi este numit **STS-l** (**Synchronous Transport Signal,** rom: semnal sincron de transport). Toate trunchiurile SONET sunt multiple de STS-1.

Primele trei coloane ale fiecărui cadru sunt rezervate pentru informaţia de administrare a sistemului, aşa cum este prezentat în fig. 2-36. Primele trei rânduri conţin informaţia suplimentară (eng: overhead) pentru secţiune; următoarele şase conţin informaţia suplimentară pentru linie. Informaţia suplimentară pentru secţiune este generată şi verificată la începutul şi la sfârşitul fiecărei secţiuni, în timp ce informaţia suplimentară pentru linie este generată şi verificată la începutul şi la sfârşitul fiecărei linii.

Un transmiţător SONET trimite cadre succesive de 810 octeţi, fără pauze între ele, chiar şi atunci când nu există date de transmis (situaţie în care trimite cadre fără semnificaţie). Din punct de vedere al receptorului, informaţia este doar un şir continuu de biţi, deci cum să ştie de unde începe un cadru? Răspunsul este că primii doi octeţi ai fiecărui cadru conţin un şablon fix pe care receptorul îl caută. Dacă găseşte şablonul în acelaşi loc la mai multe cadre consecutive, presupune ca s-a sincronizat cu transmiţătorul. Teoretic, un utilizator ar putea trimite acest şablon în informaţia utilă din cadru, dar practic acest lucru nu poate fi realizat din cauza multiplexării mai multor utilizatori în acelaşi cadru şi din alte motive.

Restul de 87 de coloane conţin 87 x 9 x 8 x 8000 = 50.112 Mbps de date ale utilizatorului. Totuşi, datele utilizatorului, numite **SPE** (**Synchronous Payload Envelope**, rom: înveliş pentru informaţie utilă sincronă) nu încep întotdeauna cu rândul 1, coloana 4. SPE poate începe oriunde în interiorul cadrului. Un pointer către primul octet este conţinut în primul rând al informaţiei suplimentare pentru linie. Prima coloană din SPE este informaţia suplimentară pentru cale (adică un antet pentru protocolul subnivelului de conexiune capăt-la-capăt).

Posibilitatea ca SPE să înceapă oriunde în cadrul SONET şi chiar să se întindă pe două cadre, aşa cum este prezentat în fig. 2-36, conferă sistemului un grad suplimentar de flexibilitate. De exemplu, dacă la sursă ajung date în timp ce se construieşte un cadru SONET gol, aceste date pot fi inserate în cadrul curent în loc să fie reţinute până la începutul următorului cadru.

3 coloane pentru

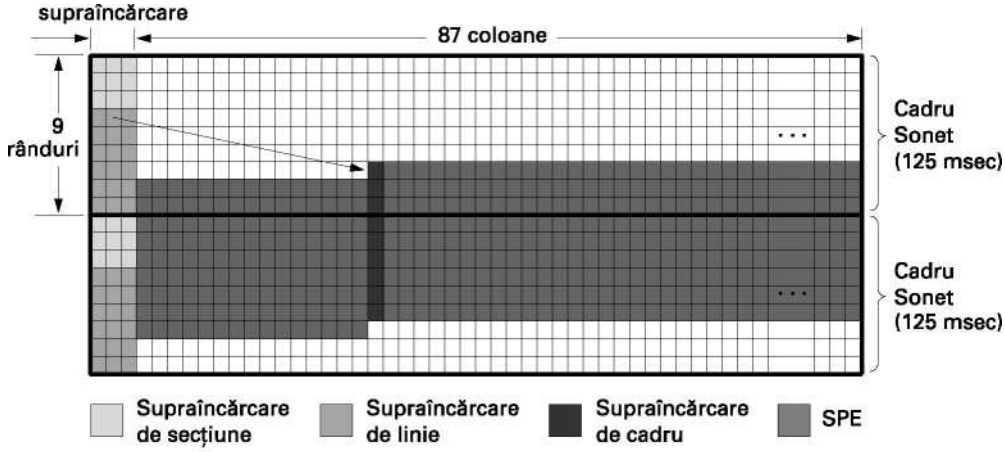


Fig. 2-36. Două cadre SONET succesive.

Ierarhia de multiplexare SONET este prezentată în fig. 2-37. Au fost definite viteze de la STS-l până la STS-192. Purtătoarea optică pentru STS-*n* este numită *OC-n* şi este identică bit cu bit cu STS-*n*, cu excepţia unui rearanjări a biţilor folosită pentru sincronizare. Numele SDH sunt diferite şi încep de la OC-3, deoarece sistemele bazate pe standardele CCITT nu au o viteză de transfer apropriată de 51.84 Mbps. Purtătorul OC-9 este prezent, deoarece este apropriat de viteza de transfer a unui trunchi de mare viteză folosit în Japonia. OC-18 şi OC-36 sunt folosite în Japonia. La calculul vitezei de transfer a datelor sunt incluse toate informaţiile suplimentare. La calculul vitezei de transfer SPE se exclud informaţiile suplimentare pentru linie şi secţiune. Pentru a determina viteza de transfer a datelor utile este exclusă orice informaţie suplimentară, fiind luate în consideraţie numai cele 86 de coloane puse la dispoziţie pentru datele utile.

Ca un comentariu suplimentar, reţineţi că atunci când o purtătoare, cum este OC-3, nu este multiplexată, dar transportă date de la o singură sursă, se adaugă la notaţie litera *c* (de la concatenare), astfel că OC-3 indică o purtătoare de 155,52 Mbps care constă din trei purtătoare OC-l diferite, în timp ce OC-3c indică o secvenţă de date de la o singură sursă la 155,52 Mbps. Cele trei secvenţe OC-l dintr-o secvenţă OC-3c sunt întreţesute pe coloane, prima coloană din secvenţa 1, apoi coloana din secvenţa 2, apoi coloana 1 din secvenţa 3, urmată de coloana 2 din secvenţa 1 şi aşa mai departe, ceea ce conduce la un cadru de 270 de coloane şi 9 rânduri.

SONET		SDH	Rata de date (Mbps)		
Electric	**Optic**	**Optic**	**Totală**	**SPE**	**Client**
STS-1	OC-1		51.8	50.112	49.536
STS-3	OC -3	STM-1	155.5	150.336	148.60
STS-9	OC -9	STM-3	466.5	451.008	445.82
STS-12	OC-12	STM-4	622.0	601.344	594.432
STS-18	OC-18	STM-6	933.1	902.016	891.648

STS-24	OC -24	STM-8	1244.1	1202.688	1188.864
STS-36	OC -36	STM-12	1866.2	1804.032	1783.296
STS-48	OC -48	STM-16	2488.3	2405.376	2377.728
STS-192	OC-192	STM-64	9953.2	9621.504	9510.912

Fig. 2-37. Ratele de multiplexare pentru SONET şi SDH.

2.5.5 Comutarea

Din punctul de vedere al unui inginer obişnuit specialist în telefonie, sistemul telefonic se împarte în două: domeniul exterior (buclele locale şi trunchiurile, deoarece ele sunt în afara oficiilor de comutare) şi domeniul interior (comutatoarele). Tocmai am aruncat o privire asupra domeniului exterior. Acum a venit timpul să examinăm domeniul interior.

În sistemul telefonic se utilizează două tehnici de comutare diferite: comutarea de circuite şi comutarea de pachete. Vom face în continuare câte o scurtă introducere pentru fiecare dintre cele două tehnici. Apoi vom studia în detaliu comutarea de circuite, acesta fiind modul de funcţionare actual al sistemul telefonic. Comutarea de pachete va fi studiată în detaliu în capitolele care urmează.

Comutarea de circuite

Atunci când formezi – tu sau calculatorul tău – un număr de telefon, echipamentele de comutare din sistemul telefonic caută o cale fizică între telefonul tău şi telefonul apelat. Această tehnică se cheamă **comutare de circuite** şi este prezentată schematic în fig. 2-38(a). Fiecare dintre cele 6 dreptunghiuri reprezintă un oficiu de comutare al companiei de telecomunicaţii (oficiu final, oficiu de taxare etc.). În acest exemplu, fiecare oficiu are trei linii de intrare şi trei linii de ieşire. Atunci când o cerere trece prin oficiu, se stabileşte (conceptual) o legătură între linia de pe care a venit cererea şi una din liniile de ieşire; aceste legături sunt reprezentate în figură de liniile punctate.

În primele zile ale telefoniei, legătura era făcută de un operator care conecta un cablu în mufele de intrare şi de ieşire. De fapt, există şi o povestioară simpatică legată de invenţia echipamentelor de comutare automată a circuitelor. Aceste echipamente au fost inventate în secolul XIX de un proprietar al unei firme de Pompe Funebre din Missouri, pe nume Almon B. Strowger. Puţin timp după ce telefonul a fost inventat, când cineva a murit, unul dintre supravieţuitori ar fi sunat operatorul oraşului şi spunând: „Vă rog, faceţi-mi legătura cu o firmă de Pompe Funebre”. Din păcate pentru domnul Strowger, existau două firme de Pompe Funebre în oraş, iar proprietarul celeilalte era soţul operatoarei telefonice a oraşului. Domnul Strowger şi-a dat repede seama că fie va inventa echipamentul telefonic de comutare automată, fie va renunţa la afacere. A ales prima opţiune. Timp de 100 de ani, echipamentele de comutare a circuitelor au fost cunoscute în toată lumea drept cutia Strowger. (Istoria nu a consemnat însă dacă operatoarea de comutare, rămasă şomeră, a obţinut ulterior un post de operator de informaţii, răspunzând la întrebări de genul „Care este numărul de telefon al unei firme de

Pompe Funebre ?”).

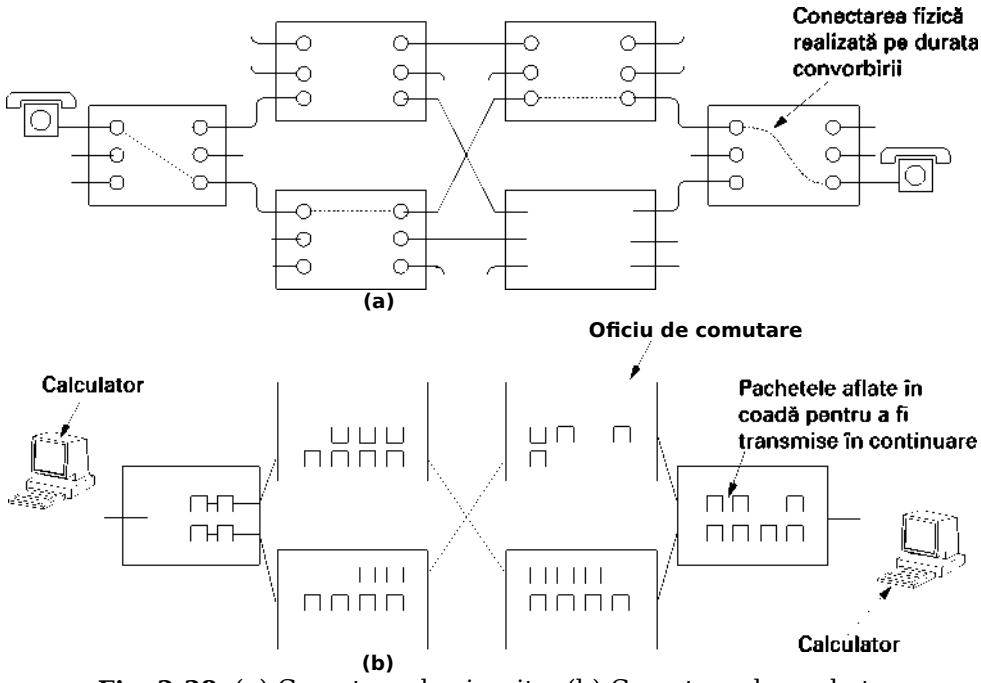


Fig. 2-38. (a) Comutare de circuite. (b) Comutare de pachete

Modelul prezentat în fig. 2-39(a) este, desigur, foarte simplificat, deoarece unele porţiuni din „drumul de cupru” între cele două telefoane pot fi, de fapt, legături prin microunde pe care sunt multiplexate mii de convorbiri. În orice caz, ideea de bază este validă: odată ce apelul a fost stabilit, există o cale dedicată între cele două capete şi va continua să existe până când convorbirea se termină.

Alternativa la comutarea circuitelor este comutarea de pachete, descrisă în fig. 2-38(b). În cazul acestei tehnologii, se vor trimite pachete individuale la cerere, fără ca o cale dedicată să fie construită în prealabil. Fiecare pachet trebuie să-şi găsească singur drumul către destinaţie.

O proprietate importantă a comutării de circuite este nevoia de a stabili o cale de la un capăt la celălalt, înainte ca datele să poată fi transmise. Intervalul de timp dintre momentul formării numărului şi până când se aude sunând telefonul apelat poate ajunge uşor la 10 sec, chiar mai mult pe distanţe mari sau în cazul convorbirilor internaţionale. În acest interval de timp, sistemul telefonic caută un drum prin cupru, după cum e prezentat în fig. 2-39(a). De remarcat că, înainte ca transmisia de date să poată începe, semnalul de apel trebuie să se propage până la destinaţie. Pentru multe aplicaţii pe calculator (de ex. verificarea creditului la punctul de vânzare), se doreşte evitarea perioadelor lungi de setare.

Ca o consecinţă a căii rezervate dintre cele două parţi care vorbesc, odată ce conexiunea a fost realizată, singura întârziere a datelor este dată de timpul de propagare a semnalului electromagnetic, de aproximativ 5 ms la 1000 Km.

Totodată, ca o consecinţă a prestabilirii traseului, pericolul de congestie dispare – aceasta înseamnă că, odată ce apelul s-a efectuat, nu vei mai primi semnalul „ocupat”. Desigur, poţi primi semnalul „ocupat” înainte de a stabili legătura, datorită imposibilităţii de comutare sau a capacităţii insuficiente a trunchiului.

Semnal de cerere convorbire

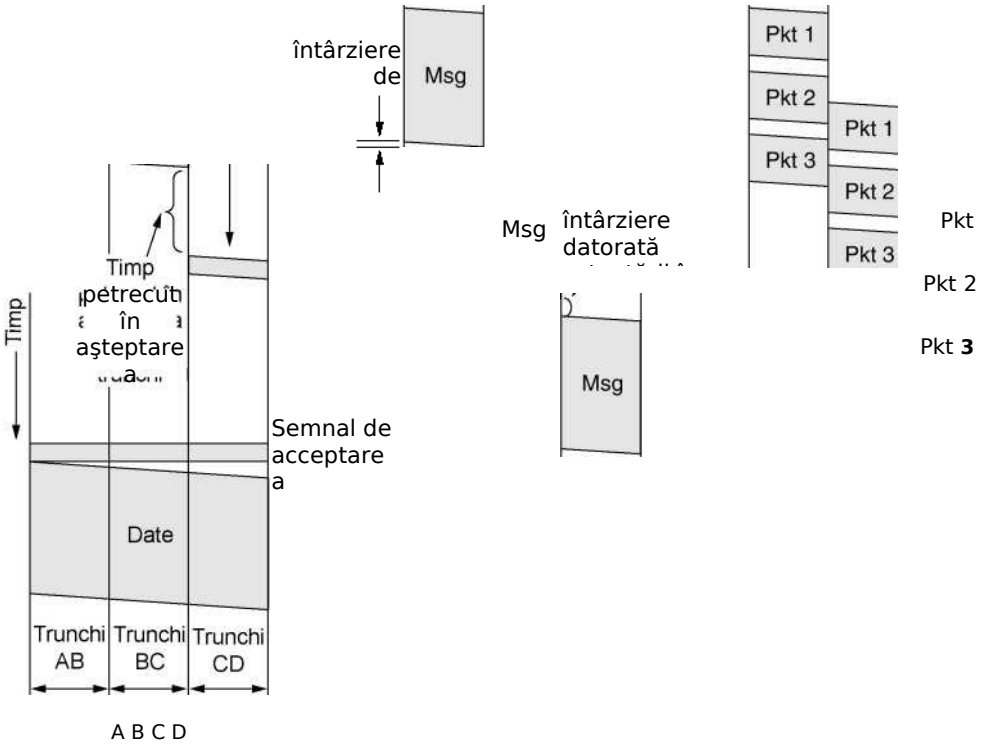


Fig. 2-39. Apariţia evenimentelor în (a) comutarea de circuite. (b) comutarea de mesaje. (c) comutarea de pachete

Comutarea de mesaje

O alternativă la strategia de comutare de circuite este **comutarea de mesaje**, prezentată în fig. 2-39(b). Atunci când se utilizează acest tip de comutare, nu se stabileşte de la început o cale între apelant şi apelat. În schimb, atunci când apelantul are de transmis un bloc de date, acesta este memorat în primul oficiu de comutare (ruter) şi este retransmis mai târziu, pas cu pas. Fiecare bloc este recepţionat în întregime, verificat pentru a detecta eventualele erori şi apoi retransmis. După cum a fost menţionat în cap. 1, o reţea care foloseşte această tehnică se numeşte reţea **memorează- şi-retransmite**.

Primul sistem de telecomunicaţie electromecanic se baza pe comutarea de mesaje şi a fost folosit pentru telegrame. Mesajul era perforat pe o bandă de hârtie la oficiul de transmisie, bandă care era apoi citită şi transmisă pe o linie de comunicaţie către următorul oficiu de pe traseu, unde era perforat pe o altă bandă de hârtie. Un operator de acolo rupea banda de hârtie şi o citea pe unul din multele cititoare de bandă hârtie, câte unul pentru fiecare magistrală de ieşire. Un astfel de oficiu de comutare era numit **oficiu de tocare a benzilor** (**torn tape office**). Benzile de hârtie nu mai se folosesc şi nici comutarea mesajelor, deci nu vom mai discuta

acest subiect în această carte.

Comutarea de pachete

În cazul comutării de mesaje, nu există nici o limită a dimensiunii blocului, ceea ce înseamnă că ruterele (din sistemele moderne) necesită discuri pentru memorarea unor blocuri mari. De asemenea, acest lucru înseamnă că un singur bloc poate ocupa o linie ruter-ruter minute întregi, comutarea de mesaje nefiind utilă pentru traficul interactiv. Pentru a rezolva aceste probleme, a fost inventată **comutarea de pachete**, aşa cum a fost descrisă în cap. 1. Reţelele cu comutare de pachete fixează o limită superioară precisă pentru dimensiunea blocului, permiţând pachetelor să fie păstrate în memoria principală a ruterului, în loc să fie salvate pe disc. Asigurându-se faptul că nici un utilizator nu va putea monopoliza o linie de transmisie mult timp (milisecunde), reţelele cu comutare de pachete au devenit adecvate pentru traficul interactiv. Un alt avantaj al comutării de pachete faţă de comutarea de mesaje este prezentat în fig. 2-39(b) şi (c): primul pachet al unui mesaj multi-pachet poate fi transmis mai departe înainte ca cel de-al doilea pachet să fie complet recepţionat, micşorând întârzierea şi îmbunătăţind productivitatea. Din aceste motive, reţelele de calculatoare folosesc, de obicei, comutarea de pachete, ocazional, comutarea de circuite, dar niciodată comutarea de mesaje.

Comutarea de circuite şi comutarea de pachete diferă în multe privinţe. Pentru început, comutarea de circuite necesită construirea unui circuit între transmiţător şi receptor înainte să înceapă comunicaţia. Comutarea de pachete nu are nevoie de pregătiri prealabile. Primul pachet poate fi trimis atunci când este disponibil.

Rezultatul unei conexiuni stabilite cu comutare de circuite este rezervarea lărgimii de bandă pe tot traseul de la transmiţător la receptor. Toate pachetele urmează această cale. Faptul că toate pachetele urmează aceeaşi cale înseamnă că nu pot ajunge la destinaţie în altă ordine decât aceea în care au fost trimise. La comutarea cu pachete nu există o cale, deci pachete diferite pot urma căi diferite, în funcţie de condiţiile de reţea din momentul în care sunt trimise. Acestea pot ajunge în altă ordine decât cea iniţială.

Comutarea cu pachete este mult mai tolerantă la erori decât comutarea cu circuite. De fapt, acesta este motivul pentru care a fost inventată. Dacă un comutator se defectează, toate circuitele care îl folosesc se termină şi nu se mai poate face trafic pe acestea. În cazul comutării cu pachete, pachetele pot fi redirecţionate astfel încât să poată ocoli comutatoarele defecte.

De asemenea, rezervarea unei căi oferă posibilitatea rezervării de lăţime de bandă. Dacă lăţimea de bandă e rezervată, atunci când un pachet ajunge la destinaţie el este transmis imediat mai departe. În cazul comutării de pachete, lăţimea de bandă nu este rezervată, şi deci pachetele ar putea aştepta până să fie trimise mai departe.

Prin rezervarea lăţimii de bandă se asigură că nu poate să apară congestie când soseşte un pachet (doar dacă apar mai multe pachete decât sunt aşteptate). Pe de altă parte, când se încearcă stabilirea unui circuit, există posibilitatea unui eşec datorat congestiei. Deci, congestia poate interveni la momente diferite în cazul comutării de circuite (la stabilirea circuitului) şi în cazul comutării de pachete (la

transmisia pachetele).

Dacă un circuit a fost rezervat pentru un anumit utilizator şi nu există trafic, lăţimea de bandă a circuitului e irosită, şi nu poate fi folosită pentru alt trafic. Comutarea de pachete nu iroseşte lăţime de bandă, deci este mai eficientă dintr-o perspectivă de ansamblu. Înţelegerea acestui compromis este crucială pentru înţelegerea diferenţei dintre comutarea de circuite şi comutarea de pachete. Compromisul este între serviciul garantat cu resurse irosite şi serviciul negarantat cu resurse neirosite.

Comutarea de pachete foloseşte transmisia de tip memorează-şi-trimite. Un pachet e salvat în memoria unui ruter, apoi trimis către următorul ruter. În cazul comutării de circuite, biţii curg continuu prin fir. În cazul comutării de pachete, transmisia memorează-şi-trimite adaugă o întârziere.

O altă diferenţă este transparenţa completă a comutării de circuite. Transmiţătorul şi receptorul pot folosi orice viteză de transfer, orice format sau orice metodă de formare a cadrului. Compania de telecomunicaţii nu cunoaşte aceste lucruri şi nici nu este interesată de ele. În cazul comutării de pachete, compania de telecomunicaţie determină parametrii de bază. O analogie grosieră ar fi o comparaţie între şosea şi calea ferată. În cazul celei dintâi, utilizatorul determină mărimea, viteza şi natura vehiculului; în cazul celei de-a doua, acest lucru îl face societatea de cale ferată. Această transparenţă face posibilă coexistenţa vocii, a faxurilor şi a datelor în sistemul telefonic.

O ultimă diferenţă între comutarea de circuite şi comutarea de pachete se referă la algoritmul de taxare. La comutarea de circuite, acesta a fost bazat de la început pe distanţă şi timp. De obicei, pentru telefoanele mobile distanţa nu contează, excepţie făcând convorbirile internaţionale, iar timpul are doar un rol minor (de exemplu un abonament cu 2000 de minute gratuite costă mai mult decât unul cu 1000 de minute gratuite, iar uneori convorbirile în timpul nopţii şi în weekend sunt mai ieftine decât în mod normal). La comutarea de pachete, timpul de conectare nu contează, dar volumul de trafic da. Pentru utilizatorii simpli, distribuitorii de Internet taxează o anumită sumă lunar pentru că e mai puţin de lucru pentru ei şi mai uşor de înţeles de către clienţi, dar reţelele de infrastructură taxează reţelele regionale pe baza volumului de trafic. Diferenţele sunt prezentate în fig. 2-40.

Criteriu	Comutarea de	Comutarea de
Realizarea conectării	Necesară	Nu e necesară
Cale fizică dedicată	Da	Nu
Fiecare pachet urmează aceiaşi	Da	Nu
Pachetele ajung în ordine	Da	Nu
Defectarea unui comutator e	Da	Nu
Banda de frecvenţă disponibilă	Fixă	Dinamică
Când poate să apară congestia	La momentul setării	La fiecare pachet
Banda de frecvenţă eventual	Da	Nu
Transmisia memorează şi	Nu	Da
Transparenţă	Da	Nu
Taxarea	Pe minut	Pe pachet

Fig. 2-40. Comparaţie între reţelele cu comutare de circuite şi cu comutare de pachete.

Datorită faptului că atât comutarea de circuite cât şi comutarea de pachete sunt foarte importante, vom reveni la ele în curând şi vom prezenta în detaliu diversele tehnologii folosite.

2.6 SISTEMUL DE TELEFONIE MOBILĂ

Sistemul tradiţional de telefonie (chiar dacă uneori înseamnă fibră optică multi-megabit), nu va putea satisface un grup tot mai mare de utilizatori: oamenii în mişcare. Oamenii se aşteaptă să efectueze convorbiri telefonice din avion, din maşină, din piscină sau în timp ce aleargă prin parc. Nu mai departe de câţiva ani se vor aştepta să trimită e-mail-uri şi să navigheze pe Internet din toate aceste locuri şi din altele. Prin urmare, există un interes extrem de mare în telefonia fără fir. În următoarele paragrafe vom studia în detaliu acest subiect.

Telefoanele fără fir există în două variante de bază: telefoanele fără fir şi telefoanele mobile (uneori numite şi **celulare**). **Telefoanele fără fir** sunt dispozitive constând dintr-o staţie bază şi un
receptor, vândute ca set pentru uzul casnic. Aceste dispozitive nu sunt folosite pentru reţele, deci nu le vom mai prezenta în continuare. În schimb ne vom concentra asupra telefoanelor mobile, care sunt folosite pentru comunicaţia de date şi voce pe arii mari.

Telefoanele mobile sunt împărţite în trei generaţii, fiecare cu o tehnologie diferită:

1. Voce analogic.
2. Voce digital.
3. Voce şi date (Internet, e-mail, etc.) digital.

Deşi mare parte a discuţiei se va referi la tehnologia acestor sisteme, e interesant de văzut în ce fel politica şi micile decizii de marketing pot avea un impact uriaş. Primul sistem mobil din SUA a fost inventat de AT&T şi extins la nivelul întregii ţări de FCC. Drept urmare, pe tot teritoriul SUA era un singur sistem (analogic), iar un telefon cumpărat în California funcţiona şi în New York. În contrast, când telefonia mobilă a ajuns în Europa, fiecare ţară şi-a inventat propriul sistem, rezultatul fiind un fiasco.

Europa a învăţat din propriile greşeli şi când s-a pus problema digitizării, guvernele s-au întâlnit şi au standardizat un singur sistem (GSM), aşa încât orice telefon mobil european să poată funcţiona oriunde în Europa. Până atunci, SUA hotărâse că guvernul nu ar trebui să intervină în procesul de standardizare, deci a lăsat digitizarea pe mâna pieţei. Această decizie a dus la telefoane diferite, fabricate de producători diferiţi. Drept urmare, SUA are două sisteme digitale mari incompatibile (plus încă unul mai mic).

În ciuda unui avans iniţial în SUA, deţinerea şi folosirea unui telefon mobil în Europa este mult mai mare ca în SUA. Crearea unui singur sistem pentru toată

Europa este unul din motive, dar mai sunt şi altele. Un al doilea domeniu unde Europa şi SUA se diferenţiază este problema atribuirii numerelor de telefon. În SUA, numerele de telefoane mobile sunt amestecate cu cele de telefoane normale (fixe). Deci, este imposibil pentru o persoană care sună , să zicem la (212) 234+5678 să ştie dacă e un post telefonic fix (mai ieftin sau gratis) sau un telefon mobil (convorbire scumpă). Pentru a preveni enervarea oamenilor la folosirea telefoanelor, companiile telefonice au decis ca posesorul telefonului mobil să plătească apelurile recepţionate. Drept urmare, mulţi oameni au ezitat sa-şi cumpere un telefon mobil de frică să nu ajungă cu o notă de plată foarte mare doar pentru că au fost sunaţi. În Europa, telefoanele mobile au un prefix special (analog numerelor cu 800 şi 900) deci sunt foarte uşor de recunoscut. Prin urmare, regula că acela care sună plăteşte se aplică şi telefoanelor mobile în Europa (cu excepţia convorbirilor internaţionale, unde costul este împărţit).

Un al treilea aspect care a avut un impact semnificativ este folosirea telefoanelor preplătite în Europa, (până la 75 % în unele zone). Acestea pot fi cumpărate din multe magazine, cu cât mai puţine formalităţi, nu mai multe decât pentru cumpărarea unui radio. Plăteşti şi pleci. Acestea sunt preîncăr- cate cu, de exemplu, 20 sau 50 Euro şi pot fi reîncărcate (folosind un cod PIN secret) când suma se termină. În consecinţă, practic fiecare adolescent şi mulţi dintre copiii mici din Europa au telefoane mobile (de obicei preîncărcate), pentru ca părinţii să îi poată găsi, fără să existe pericolul unei note de plată imense pentru telefonul copilului. Dacă telefonia mobilă e folosită doar ocazional, este practic gratuită, din moment ce nu există o taxă lunară sau taxare pentru apelurile recepţionate.

2.6.1 Prima generaţie de telefoane mobile: Voce analogică

Am discutat suficient despre politica şi aspectele de marketing ale telefoanelor mobile. Acum să ne uităm la tehnologie, pornind cu primele sisteme. Radiotelefoanele mobile au fost folosite sporadic, pentru comunicaţii maritime şi militare, încă din timpul primelor decenii ale secolului XX. În 1946, primul sistem de telefoane pentru automobile a fost pus în funcţiune în St. Louis. Acest sistem folosea un singur emiţător amplasat pe o clădire înaltă şi avea un singur canal folosit atât pentru emisie cât şi pentru recepţie. Pentru a vorbi, un utilizator trebuia să apese un buton care activa emiţătorul şi dezactiva receptorul. Astfel de sisteme, cunoscute sub denumirea de **sisteme cu buton de emisie** au fost instalate în câteva oraşe, spre sfârşitul anilor 1950. Radioul CB, taxiurile şi maşinile de poliţie folosesc deseori această tehnologie.

În 1960 a fost instalat **IMTS** (**Improved Mobile Telephone System**, rom: sistem îmbunătăţit de telefonie mobilă). Şi acesta foloseşte un emiţător de mare putere (200 W), amplasat pe vârful unui deal, dar utilizează două frecvenţe: una pentru emisie şi una pentru recepţie. Prin urmare, nu mai este nevoie de butonul de emisie. Deoarece toată comunicaţia dinspre telefoanele mobile se desfăşoară pe un canal diferit de cel pe care telefoanele ascultă, utilizatorii telefoanelor mobile nu se mai pot auzi unii pe alţii (spre deosebire de sistemul cu buton de emisie folosit la

taxiuri).

IMTS suportă 23 de canale distribuite între 150 MHz şi 450 MHz. Din cauza numărului mic de canale, utilizatorii trebuie să aştepte deseori perioade lungi de timp până când obţin tonul. De asemenea, datorită puterii mari a emiţătorului aflat la înălţime, sistemele adiacente trebuie să se afle la câteva sute de kilometri distanţă, pentru a se evita interferenţa. În concluzie, sistemul a fost impracticabil datorită posibilităţilor limitate.

Sistemul avansat de telefonie mobilă

Totul s-a schimbat odată cu **AMPS (Advanced Mobile Phone System**, rom: sistem avansat de telefonie mobilă), inventat de Bell Labs şi instalat pentru prima dată în S.U.A. în 1982. Sistemul a fost folosit şi în Anglia, unde purta denumirea TACS şi în Japonia, unde se numea MCS-L1. Deşi nu mai este la modă, îl vom studia mai în detaliu, deoarece multe din proprietăţile sale fundamentale au fost moştenite de succesorul sau digital, D-AMPS, din considerente de compatibilitate.

În toate sistemele de telefonie mobilă, o regiune geografică e împărţită în **celule** şi de aceea telefoanele sunt uneori numite telefoane celulare. În AMPS, o celula are de obicei 10 până la 20 km lăţime; în sistemele digitale celulele sunt mai mici. Fiecare celulă foloseşte un set de frecvenţe, nefolosit de nici unul dintre vecinii săi. Ideea de bază care conferă sistemelor celulare o capacitate semnificativ mai mare decât a tuturor sistemelor anterioare, constă în folosirea de celule relativ mici şi refolosirea frecvenţelor de transmisie în celule apropiate (dar nu adiacente). În timp ce într-un sistem IMTS de 100 km lăţime poate exista un singur apel pe fiecare frecvenţă, un sistem AMPS poate avea 100 de celule de 10 km în aceeaşi regiune şi este capabil să suporte 10 până la 15 apeluri pe fiecare frecvenţă, în celule separate de distanţe mari. Astfel, proiectarea celulară determină creşterea capacităţii sistemului cu cel puţin un ordin de mărime, cu atât mai mult cu cât celulele devin mai mici. Mai mult, celulele de dimensiuni reduse necesită puteri mici, ceea ce implică folosirea de transmiţătoare mai ieftine şi de dimensiuni mai mici. Telefoanele de mână emit 0,6W; emiţătoarele de pe maşini au de obicei 3 W, valoarea maximă permisă de FCC.

Ideea refolosirii frecvenţelor este ilustrată în fig. 2-41(a). În mod normal, celulele sunt aproximativ circulare, dar ele pot fi modelate mai uşor ca hexagoane. În fig. 2-41(a), celulele au toate aceeaşi dimensiune. Ele sunt grupate împreună în unităţi de 7 celule. Fiecare literă indică un grup de frecvenţe. Trebuie remarcat că pentru fiecare set de frecvenţe există o zonă tampon, de lăţime aproximativ egală cu dublul mărimii unei celule, în care acea frecvenţa nu este refolosită, realizând astfel o delimitare mai bună şi o interferenţă scăzută.

O problemă majoră o constituie găsirea locurilor înalte pentru instalarea antenelor staţiei de bază. Această problemă a determinat pe unii furnizori de servicii de telecomunicaţii să încheie contracte cu Biserica Romano-catolică pentru că aceasta dispune de un număr substanţial de potenţiale locuri înalte pentru antene, toate aflate, în mod convenabil, sub o singură administraţie.

Într-o zonă în care numărul de utilizatori s-a mărit atât de mult încât sistemul a

devenit supraîncărcat, se reduce puterea, iar celulele supraîncărcate se divizează în **microcelule,** pentru a permite mai multe refolosiri de frecvenţe, aşa cum este arătat în fig. 2-41(b). Determinarea dimensiunii maxime a celulelor constituie o problemă complexă şi este tratată în (Hac, 1995).

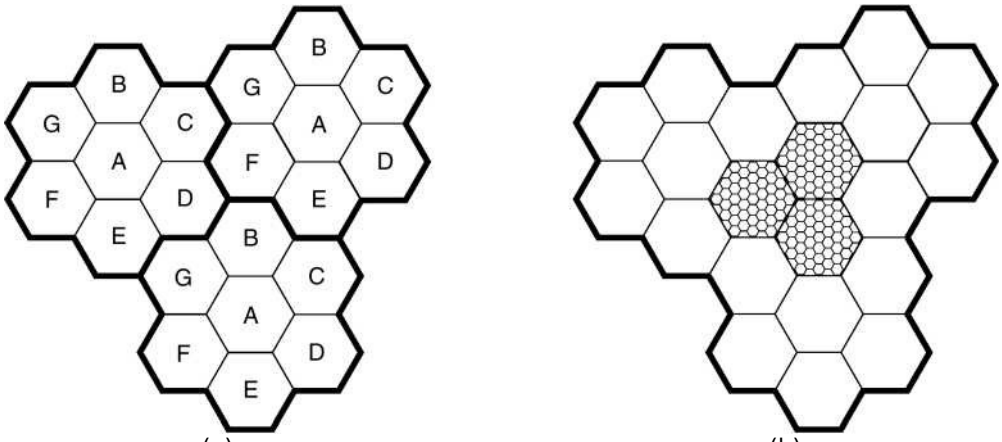


Fig. 2-41. (a) Frecvenţele nu sunt refolosite în celule adiacente. (b) Pentru a adăuga mai mulţi utilizatori se pot folosi celule mai mici.

În centrul fiecărei celule se află o staţie de bază către care transmit toate telefoanele din celulă. Staţia de bază cuprinde un calculator şi un emiţător/receptor conectat la o antenă. Într-un sistem de dimensiuni reduse, toate staţiile de bază sunt conectate la un singur dispozitiv, denumit **MTSO** (**Mobile Telephone Switching Office**, rom: oficiu de comutare pentru telefonie mobilă) sau **MSC** (**Mobile Switching Center**, rom: centru de comutare mobil). Într-un sistem de dimensiuni mari, pot fi necesare mai multe MTSO-uri, toate acestea conectându-se la un al doilea nivel MTSO şi aşa mai departe. MTSO-urile sunt în esenţă oficii finale – ca şi în sistemul telefonic – şi sunt de fapt conectate la cel puţin un oficiu final din sistemul telefonic. MTSO-urile comunică între ele, cu staţiile de bază şi cu PSTN-ul folosind o reţea cu comutare de pachete.

În orice moment, orice telefon mobil se află, în mod logic, într-o anumită celulă şi sub controlul staţiei de bază a celulei respective. Când un telefon mobil părăseşte o celulă, staţia sa de bază sesizează o scădere a semnalului dinspre telefon şi întreabă toate staţiile de bază înconjurătoare cât de puternic este semnalul pe care îl recepţionează de la respectivul telefon. Staţia de bază transferă apoi proprietatea asupra telefonului către celula care recepţionează cel mai puternic semnal, aceasta fiind şi celula în care se află acum telefonul. Telefonul este informat despre noul său şef, iar dacă un apel este în derulare în acel moment, telefonul va fi rugat să comute pe un canal nou (deoarece canalul vechi nu este refolosit în nici o celulă adiacentă). Acest proces poartă denumirea de **timp mort** şi durează aproximativ 300 ms. Atribuirea canalului se face de către MTSO, care este centrul nervos al sistemului. Staţiile de bază sunt, de fapt, doar nişte radio-relee.

Timpii morţi pot fi eliminaţi în două feluri. Într-un **timp mort soft** telefonul e luat în primire de noua staţie de bază înainte ca staţia veche să îl cedeze. Astfel, nu apare nici o discontinuitate. Partea proastă a acestei variante este că telefonul trebuie să fie capabil să se seteze pe două frecvenţe în acelaşi timp (cea veche şi cea nouă). Nici telefoanele de primă generaţie, nici cele de generaţia a doua nu pot face acest lucru.

Într -un **timp mort hard**, vechea staţie de bază deconectează telefonul înainte ca staţia nouă să îl preia. Dacă noua staţie de bază nu poate să preia telefonul (de ex. pentru că nu există nici o frecvenţă disponibilă), convorbirea se termină brusc. Utilizatorii constată acest lucru, dar câteodată situaţia este inevitabilă din cauza modului în care sunt proiectate telefoanele actuale.

Canale

Sistemul AMPS utilizează 832 canale full-duplex, fiecare constând dintr-o pereche de canale simplex. Există astfel 832 canale simplex pentru transmisie de la 824 la 849 MHz şi 832 canale simplex pentru recepţie de la 869 la 894 MHz. Fiecare din aceste canale simplex are o lăţime de 30 KHz. Din această cauză AMPS foloseşte FDM pentru a separa canalele.

În banda de 800 MHz, undele radio au aproximativ 40 cm lungime şi se propagă în linie dreaptă. Ele sunt absorbite de copaci şi plante şi sunt deviate de pământ şi clădiri. Este posibil ca un semnal emis de un telefon mobil să ajungă la staţia de bază pe calea directă, dar tot la fel de bine poate să ajungă puţin mai târziu, după ce este deviat de pământ sau clădiri. Aceasta poate să conducă la un efect de ecou sau la distorsionarea semnalului (atenuare multi-căi). Uneori este posibil să se audă chiar şi o convorbire îndepărtată care a suferit mai multe deviaţii. Cele 832 de canale se împart în patru categorii:

1. Comandă (baza către mobil) pentru gestionarea sistemului.
2. Semnalizare (baza către mobil) pentru a anunţa utilizatorii de telefoane mobile că sunt apelaţi.
3. Acces (bidirecţional) pentru stabilirea apelului şi alocarea canalului.
4. Date (bidirecţional) pentru voce, fax sau date.

Pentru comenzi sunt rezervate douăzeci şi unu de canale şi acestea sunt fixate în fiecare telefon într-un PROM. Deoarece aceleaşi frecvenţe nu pot fi refolosite în celule învecinate, numărul real de canale de voce disponibile pe celulă este mult mai mic decât 832, de regulă 45.

Gestiunea apelului

Fiecare telefon mobil din AMPS are un număr serial pe 32 biţi şi un număr de telefon de 10 cifre în PROM-ul propriu. Numărul de telefon este format dintr-un cod al zonei de 3 cifre pe 10 biţi şi un număr de abonat de 7 cifre pe 24 de biţi. Atunci când este activat, un telefon scanează o listă prepro- gramată cu 21 canale de comandă, pentru a descoperi semnalul cel mai puternic.

Apoi telefonul difuzează propriul număr serial de 32 de biţi şi numărul de telefon de 34 de biţi. Ca orice altă informaţie de comandă din AMPS, acest pachet

este transmis în formă digitală, de mai multe ori şi cu un cod corector de erori, deşi canalele de voce sunt analogice.

Atunci când staţia de bază aude anunţul, sesizează MTSO-ul care înregistrează existenţa noului său client şi informează de asemenea MTSO-ul clientului asupra poziţiei sale curente. În timpul unei funcţionări normale, telefonul mobil se reînregistrează la fiecare aproximativ 15 minute.

Pentru a face un apel, un utilizator de telefon mobil activează telefonul, introduce de la taste numărul de apelat şi apasă butonul SEND. Telefonul transmite apoi numărul de apelat şi identitatea proprie pe canalul de acces. Dacă pe canalul de acces apare o coliziune, se încearcă din nou mai târziu. Atunci când primeşte o cerere, staţia de bază informează MTSO-ul. Dacă apelantul este un client al companiei MTSO (sau unul din parteneri), MTSO-ul caută un canal liber pentru apel. Dacă

se găseşte unul, numărul canalului este transmis înapoi pe canalul de comandă. Telefonul mobil comută apoi automat pe canalul de voce selectat şi aşteaptă până când partea apelată ridică telefonul.

Apelurile primite acţionează diferit. La început, toate telefoanele libere ascultă continuu canalul de semnalizare (paging) pentru a detecta mesajele adresate lor. Atunci când se face apel către un telefon mobil (fie de la un telefon fix, fie de la un alt telefon mobil), se transmite un pachet către MTSO-ul apelatului pentru a descoperi unde se află acesta. Se transmite apoi un pachet către staţia de bază din celula sa curentă, care transmite apoi pe canalul de semnalizare (paging) un mesaj de difuzare de forma următoare: „Unitatea 14, eşti acolo?”. Telefonul apelat răspunde apoi cu „Da” pe canalul de acces. Baza spune apoi ceva de genul „Unitatea 14, ai un apel pe canalul 3”. În acest moment, telefonul apelat comută pe canalul 3 şi începe să sune (sau cântă o melodie pe care proprietarul a primit-o cadou de ziua lui).

2.6.2 A doua generaţie de telefoane mobile: Voce digitală

Prima generaţie de sisteme celulare a fost analogică. Cea de-a doua generaţie este digitală. Aşa cum nu a existat o standardizare la prima generaţie, nu a existat nici la a doua. Patru sisteme se folosesc în prezent. D-AMPS, GSM, CDMA şi PDC. Mai jos le vom discuta pe primele trei. PDC e folosit doar în Japonia şi este practic D-APMS modificat pentru a respecta compatibilitatea cu sistemul analogic de primă generaţie din Japonia. Numele de **PCS** (**Personal Communications Services**, rom: serviciu de comunicaţii personale) este folosit câteodată în literatura de marketing pentru a indica un sistem de generaţia a doua (adică digital). Iniţial se referea la un telefon mobil care folosea banda de 1900 MHz, dar distincţia nu se mai face acum.

D-AMPS – Sistem digital avansat de telefonie mobilă

A doua generaţie a sistemelor AMPS este **D-AMPS (The Digital Advanced Mobile Phone System**, rom: sistem digital avansat de telefonie mobilă) şi este complet digital. Este descris în Standardul Internaţional IS-54 şi în succesorul acestuia, IS-136. D-AMPS a fost proiectat cu atenţie pentru a coexista cu AMPS,

aşa încât ambele generaţii de telefoane mobile , şi cele de primă generaţie şi cele de a doua, să poată opera simultan în aceeaşi celulă. D-AMPS foloseşte aceleaşi canale de 30KHz ca şi AMPS şi aceleaşi frecvenţe, astfel încât un canal poate fi analogic şi cele adiacente digitale. În funcţie de ponderile tipurilor telefoanelor dintr-o celulă, MTSO-ul celulei determină care canale sunt analogice şi care sunt digitale şi poate modifica dinamic tipul canalului, după cum se schimbă ponderile telefoanelor în celulă.

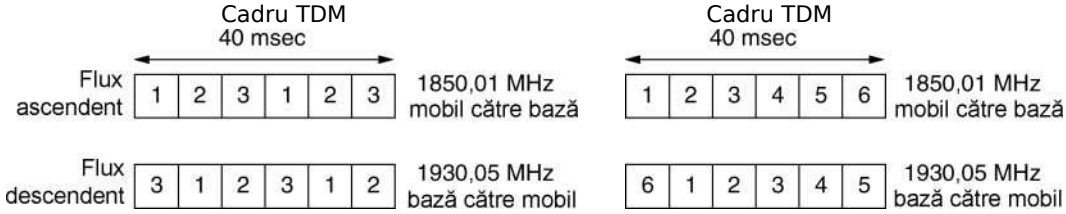
Când D-AMPS a fost introdus ca serviciu, s-a adăugat o nouă bandă de frecvenţă disponibilă pentru a gestiona încărcarea care se aştepta să crească. Canalele ascendente (de transmisie) erau în domeniul de frecvenţe 1880-1910 MHz, iar cele descendente (de recepţie) corespunzătoare erau în domeniul 1930-1990 MHz, organizate tot în perechi, ca şi la AMPS. În această bandă, undele au 16 cm lungime, aşa că antena standard % din lungimea de undă este de numai 4 cm, rezultând astfel telefoane mai mici. Oricum, multe telefoane D-AMPS pot folosi atât banda de 850 MHz cât şi pe cea de 1900 MHz, pentru a avea un domeniu mai mare de canale disponibile.

Pe un telefon mobil D-AMPS, semnalul de voce captat de microfon este digitizat şi compresat folosind un model mai complicat decât varianta cu modulaţie delta şi codificare predictivă studiată mai devreme. Compresia ia în considerare proprietăţi detaliate ale vocii umane pentru a transforma lărgimea de bandă standard de la codarea PCM (56 Kbps) la 8 Kbps sau mai puţin. Compresia este realizată de un circuit numit **vocoder** (Bellamy,2000). Compresia este făcută în telefon, şi nu în staţia

de bază sau la oficiul final, pentru a se reduce numărul de biţi trimişi prin aer. În cazul telefoniei fixe, nu există nici un beneficiu dacă se face compresia în telefon, pentru că reducerea traficului pe bucla locală nu măreşte deloc capacitatea sistemului.

La telefonia mobilă se câştigă atât de mult prin digitizarea şi compresia în telefon, încât la DAMPS trei utilizatori pot împărţi o pereche de frecvenţe folosind multiplexarea prin divizare în timp. Fiecare pereche de frecvenţe suportă 25 cadre/sec, adică 40 ms pentru fiecare cadru. Fiecare cadru este împărţit în şase intervale de timp de câte 6,67 ms, după cum se arată în fig. 2-42(a) pentru cea mai joasă pereche de frecvenţe.

Fiecare cadru cuprinde trei utilizatori, care stau la rând pentru a transmite şi a recepţiona. În timpul intervalului 1 din fig. 2-42 (a), de exemplu, utilizatorul 1 poate transmite către staţia de bază şi utilizatorul 3 recepţionează de la staţia de bază. Fiecare interval cuprinde 324 de biţi, dintre care 64 sunt folosiţi pentru timpii de gardă, sincronizare şi comandă, lăsând 260 de biţi pentru informaţia utilă utilizatorului. Dintre biţii de informaţie utilă, 101 sunt folosiţi pentru corectarea erorilor datorate mediului aerian perturbat şi în final rămân doar 159 de biţi pentru semnalul de voce comprimat. Având 50 de intervale/sec, lăţimea de bandă disponibilă pentru semnalul de voce comprimat este puţin sub 8 Kbps, adică 1/7 din lărgimea de bandă standard de la PCM.


Cadru TDM
40 msec
Flux ascendent
1 2 3 1 2 3
1850,01 MHz mobil către bază
Flux descendent
3 1 2 3 1 2
1930,05 MHz bază către mobil
Cadru TDM
40 msec
1 2 3 4 5 6
1850,01 MHz mobil către bază
6 1 2 3 4 5
1930,05 MHz bază către mobil

interval (secvenţă) de 324 de biţi:
64 de biţi de comandă
101 biţi pentru corectarea erorilor
159 biţi pentru datele corespunzătoare convorbirii

(a) (b)

Fig. 2-42. (a) Un canal S-AMPS cu trei utilizatori. (b) un canal D-AMPS cu şase utilizatori

Folosind algoritmi de compresie mai buni este posibil să se reducă datele de voce la 4 Kbps, caz în care şase utilizatori pot fi cuprinşi într-un cadru, după cum se arată în fig. 2-42(b). Din punctul de vedere al operatorului, posibilitatea de a înghesui de trei sau de şase ori mai mulţi utilizatori D-AMPS în acelaşi spectru dedicat unui singur utilizator de AMPS este un câştig uriaş şi aşa se explică popularitatea PCS-ului. Bineînţeles, calitatea vorbirii la 4 Kbps nu se compară cu cea atinsă la 56 Kbps, dar puţini operatori PCS îşi fac reclame cu referire la calitatea sunetului. De asemenea, ar trebui să fie clar că, pentru date, un canal de 8 Kbps nu este nici măcar la fel de bun ca un antic modem de 9600 bps.

Structura de comandă a D-AMPS este destul de complicată. Descriere sumară:

grupuri de 16 cadre formează un supercadru, cu informaţii de comandă specifice prezente în fiecare supercadru de un număr limitat de ori. Şase canale principale de comandă sunt folosite pentru configurarea sistemului, comanda în timp real sau offline, semnalizare, răspunsul la cererea de acces şi mesajele scurte. În principiu, funcţionează ca un AMPS. Când un mobil este deschis, el contactează staţia de bază, pentru a-şi anunţa apariţia, şi apoi ascultă pe un canal de comandă eventualele apeluri. Când recepţionează un mobil nou, MTSO informează baza utilizatorului despre locul unde se află acesta, pentru a se putea ruta corect convorbirile.

O diferenţă între AMPS şi D-AMPS este modul în care se tratează timpii morţi. În AMPS, MTSO se descurcă singur, fără nici un ajutor de la dispozitivul mobil. După cum se poate vedea în fig. 2-42, în D-AMPS, 1/3 din timp telefonul nici nu trimite nici nu primeşte. Ele folosesc acest timp pentru a măsura calitatea liniei. Când descoperă că semnalul începe să slăbească, anunţă MTSO, care apoi poate întrerupe conexiunea, moment în care mobilul poate încerca să se conecteze la o altă staţie de bază cu un semnal mai puternic. La fel ca la AMPS, acest timp mort durează totuşi 300msec. Această tehnică se numeşte **MAHO (Mobile Assisted HandOff**, rom: timpi morţi asistaţi de mobil).

GSM – Sistemul global pentru comunicaţii mobile

D-AMPS este larg folosit în SUA şi (într-o formă modificată) în Japonia. Practic, în rest, peste tot în lume se foloseşte un sistem numit **GSM** (**Global System for Mobile Communications**, rom: sistemul global pentru comunicaţii mobile), care a început să fie folosit şi în SUA la o scară limitată. La o privire generală, GSM este similar D-AMPS-ului. Amândouă sunt sisteme celulare. În ambele sisteme se foloseşte multiplexarea prin divizarea de frecvenţă, fiecare mobil transmiţând pe o frecvenţă şi recepţionând pe una mai ridicată (cu 80 MHz mai mare pentru D-AMPS şi cu 55 MHz mai mare pentru GSM). De asemenea, în ambele sisteme se foloseşte multiplexarea cu divizare în timp pentru a împărţi o singură pereche de frecvenţe în intervale de timp folosite în comun de mai multe dispozitive mobile. Oricum, canalele GSM sunt mult mai largi decât canalele AMPS (200kHz faţă de 30 KHz) şi servesc doar câţiva utilizatori în plus (8 faţă de 3), permiţând GSM-ului o rată de transmisie per utilizator mult mai mare decât la D-AMPS.

Mai jos vom discuta pe scurt unele dintre proprietăţile principale ale GSM. Oricum, standardul GSM are peste 5000 [sic] de pagini. O mare parte din acest material se referă la aspecte inginereşti ale sistemului, în special la proiectarea receptoarelor pentru a fi capabile să gestioneze propagarea semnalelor pe mai multe căi, şi sincronizarea transmiţătoarelor şi receptoarelor. Nici unul dintre acestea nu va fi menţionat în continuare.

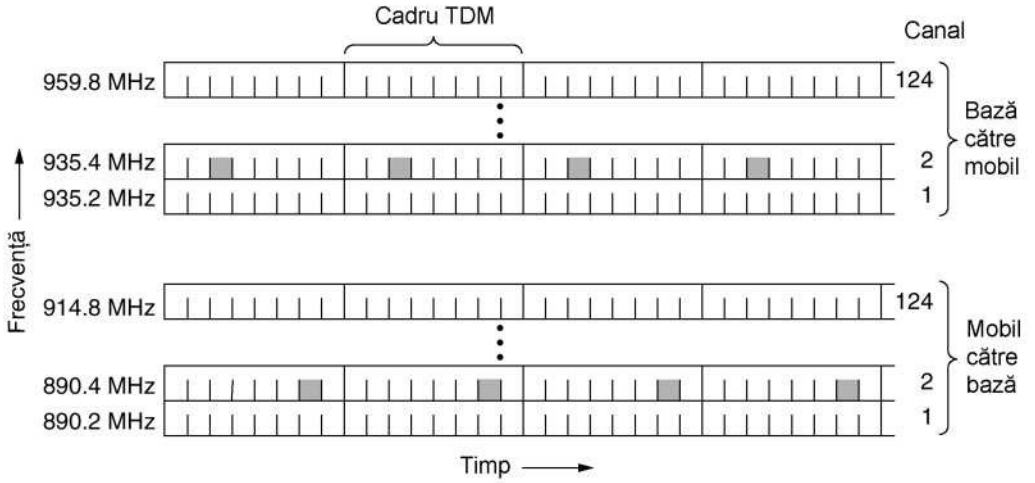


Fig. 2-43. GSM foloseşte 124 canale de frecvenţă, fiecare dintre ele folosind un sistem TDM cu opt intervale

Fiecare bandă de frecvenţă are 200 KHz, după cum se arată în fig. 2-43. Un sistem GSM are 124 de perechi de canale simplex. Fiecare canal simplex, are 200 KHz şi suportă opt conexiuni diferite, folosind multiplexarea prin divizarea în timp. Fiecărei staţii active la momentul respectiv i se atribuie un interval de timp dintr-o pereche de canale. Teoretic, pot fi suportate 992 de canale în fiecare celulă, dar multe dintre ele nu sunt disponibile, pentru a se evita conflictele de frecvenţă, cu celulele vecine. În fig. 2-43, toate cele opt intervale de timp marcate aparţin aceleiaşi conexiuni, câte patru dintre ele în fiecare direcţie. Transmisia şi recepţia nu se fac în acelaşi interval de timp pentru că aparatele GSM nu pot trimite şi primi în acelaşi timp şi durează până se face comutarea între emisie şi recepţie. Dacă staţia mobilă asociată frecvenţei 890.4/935.4 MHz şi intervalului de timp 2 ar vrea să transmită la staţia de bază, trebuie să folosească acele patru intervale de timp inferioare (şi pe cele care le urmează cronologic), punând informaţie în fiecare interval până când toată informaţia este transmisă.

Intervalele TDM prezentate în fig. 2-43 fac parte dintr-o ierarhie complexă de cadre. Fiecare interval TDM are o structură specifică, iar grupurile de intervale TDM formează multicadre, şi acestea având o structură specifică. O variantă simplificată a acestei ierarhii este prezentată în fig. 2-44. Aici putem vedea că fiecare interval TDM constă dintr-un cadru de 148 de biţi care ocupă canalul pentru 577 ps (incluzând şi un timp de gardă de 30 ps după fiecare interval). Fiecare cadru de date începe şi se termină cu trei biţi de 0, pentru aliniere. De asemenea conţine două câmpuri de *informaţie* de 57 de biţi, fiecare având un bit de comandă care spune dacă următorul cadru va fi de date sau de voce. Între câmpurile de *informaţie* se află un câmp de 26 de biţi de *sincronizare* (antrenare) care este folosit de receptor pentru sincronizarea la extremităţile cadrului de la transmiţător.

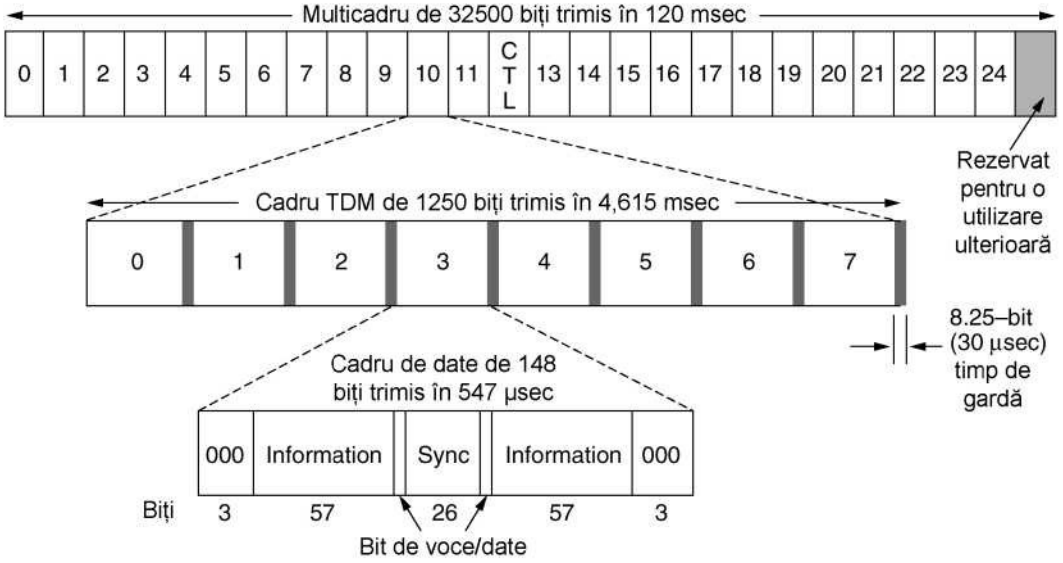

Multicadru de 32500 biți trimis în 120 msec
0 1 2 3 4 5 6 7 8 9 10 11 CTL 13 14 15 16 17 18 19 20 21 22 23 24
Rezervat pentru o utilizare ulterioară
Cadru TDM de 1250 biți trimis în 4,615 msec
0 1 2 3 4 5 6 7
8.25–bit (30 µsec) timp de gardă
Cadru de date de 148 biți trimis în 547 µsec
000 Information Sync Information 000
Biți 3 57 26 57 3
Bit de voce/date

Fig. 2-44. Un fragment din structura de cadre GSM

Un cadru de date e transmis în 547 psec, dar un transmiţător are voie să trimită câte un cadru de date la fiecare 4,615 msec, deoarece este pe acelaşi canal cu alte şapte staţii. Rata totală a fiecărui canal este de 270.833 bps, împărţită între opt utilizatori. Aceasta înseamnă 33.854 Kbps în total, mai mult decât dublu faţă de D-AMPS, 324 de biţi de 50 de ori pe secundă, adică 16,2 Kbps. Oricum, ca şi la APMS, informaţia suplimentară consumă o mare parte din lărgimea de bandă, lăsând 24,7 Kbps fiecărui utilizator pentru informaţia utilă, înainte de corectarea erorilor. După corectarea erorilor, 13 Kbps rămân pentru voce, rezultând o calitate mult mai bună decât la D-AMPS (dar folosind mai multă lăţime de bandă).

După cum se poate vedea în fig. 2-44, opt cadre de date formează un cadru TDM şi 26 cadre TDM formează un multicadru de 120 msec. Dintre cele 26 de cadre TDM dintr-un multicadru, cadrul 12 e folosit pentru comandă, iar cadrul 25 e rezervat pentru o utilizare ulterioară, deci numai 24 sunt disponibile efectiv pentru traficul utilizatorilor.

Oricum, pe lângă multicadrul de 26 de intervale din fig. 2-44, se foloseşte un multicadru de 51 de intervale (nu este desenat). Unele dintre aceste intervale sunt folosite pentru a menţine câteva canale de comandă folosite pentru gestiunea sistemului. **Canalul de difuzare a comenzii** este un flux continuu de ieşire de la staţia de bază, care conţine elemente pentru identificarea staţiei şi starea canalului. Toate staţiile mobile îşi monitorizează puterea semnalului, pentru a constata dacă s-au mutat într-o celulă nouă.

Canalul dedicat de comandă este folosit pentru actualizarea locaţiei, înregistrarea şi setările convorbirii. În particular, fiecare staţie de bază reţine o bază de date a staţiilor mobile care se află sub jurisdicţia sa. Informaţiile necesare pentru a menţine această bază de date se trimit pe canalul dedicat de comandă.

În sfârşit, există canalul comun de comandă, care este împărţit în trei subcanale

logice. Primul dintre ele este canalul de semnalizare, pe care staţiile de bază îl folosesc pentru a anunţa apelurile primite. Fiecare mobil monitorizează continuu acest canal pentru a vedea dacă sunt convorbiri la care trebuie să răspundă. Al doilea este canalul de acces aleator, care le permite utilizatorilor să ceară un loc pe canalul dedicat de comandă. Dacă două cereri intră în coliziune, ele sunt bruiate şi ambele trebuie să încerce mai târziu. Folosind canalul dedicat de comandă, staţiile pot iniţializa un apel. Intervalul primit este anunţat prin cel de al treilea subcanal, **canalul de acces permis**.

CDMA – Acces multiplu cu divizarea codului

D-AMPS şi GSM sunt sisteme convenţionale. Amândouă folosesc FDM şi TDM pentru a împărţi spectrul în canale şi canalele în intervale de timp. Mai există totuşi un al treilea tip, **CDMA** (**Code Division Multiple Acces**, rom: acces multiplu cu divizarea codului) care funcţionează complet diferit. Când CDMA a fost propus prima dată, industria i-a răspuns cu aceeaşi atitudine cu care regina Isabella i-a răspuns lui Columb când acesta i-a propus să ajungă în India navigând în direcţia opusă. Totuşi, prin insistenţa unei singure companii, Qualcomm, CDMA s-a maturizat într-atât încât nu este doar acceptat, ci este acum văzut ca fiind cea mai bună soluţie tehnică disponibilă şi de bază pentru sistemele mobile de generaţia a treia. De asemenea, în SUA se foloseşte pe scară largă în sistemele de generaţia a doua, concurând cu D-AMPS. Spre exemplu, Sprint PCS foloseşte CDMA, iar AT&T Wireless foloseşte D-AMPS. CDMA este descris în standardul internaţional IS-95 şi uneori este referit cu acest nume. Numele de marcă **cdmaOne** este de asemenea folosit.

CDMA e complet diferit de AMPS, D-AMPS şi GSM. În loc să se împartă domeniul de frecvenţe disponibil în câteva sute de canale mici, CDMA permite fiecărei staţii să transmită tot timpul în tot spectrul. Mai multe transmisii simultane sunt separate folosind teoria codării. CDMA renunţă la ideea că toate cadrele care intră în coliziune sunt total bruiate. În schimb, mai multe semnale recepţionate vor fi sumate liniar.

Înainte de a prezenta algoritmul, să considerăm analogia cu o sală de aşteptare a unui aeroport cu mai multe perechi de oameni discutând. TDM ar însemna că toţi oamenii sunt în centrul încăperii, dar vorbesc pe rând. FDM ar însemna că fiecare om este într-un colţ, fiecare vorbeşte în acelaşi timp cu ceilalţi, dar independent de ei. CDMA ar însemna că fiecare persoană este în centrul încăperii, toţi vorbesc în acelaşi timp, dar fiecare pereche de vorbitori conversează într-o limbă diferită. Cei ce vorbesc în franceză ascultă doar ceea ce este în franceză, ignorând orice nu este în limba franceză. Astfel, cheia CDMA este posibilitatea de a extrage semnalul dorit şi de a respinge restul de semnale. O descriere simplificată a CDMA este prezentată mai jos.

În CDMA durata fiecărui bit este divizată în *m* intervale scurte numite **felii** (eng.: **chips**). În mod normal sunt 64 sau 128 de felii pe bit, dar în exemplul de mai jos vom folosi doar opt pentru simplificare. Fiecare staţie are un cod unic pe *m biţi* numit **secvenţă de felii** (**chip sequence**). Pentru a transmite un bit 1, o staţie trimite propria secvenţă de felii. Pentru a transmite un bit 0, trimite complementul

faţă de unu al secvenţei sale de felii. Nici un alt şablon nu este permis. Deci, pentru *m*=8, dacă staţiei A îi este asociată secvenţa 00011011, ea trimite un bit 1 trimiţând 0001011 şi un bit 0 trimiţând 11100100.

Creşterea cantităţii de informaţie de trimis de la *b* biţi/sec la *mb* felii/sec se poate face doar dacă lărgimea de bandă disponibilă creşte de *m* ori, făcând din CDMA o formă de comunicaţie cu spectru larg, (presupunând că nu se fac schimbări în tehnicile de modulaţie sau codificare). Dacă avem o bandă de 1MHz pentru 100 de staţii, folosind FDM fiecare ar avea 10 KHz şi ar trimite 10Kbps (presupunând un bit pe Hz). Cu CDMA, fiecare staţie foloseşte toată banda de 1 MHz, deci rata feliilor este de 1 Mega pe secundă. Cu mai puţin de 100 de felii pe bit, lărgimea de bandă efectivă pentru fiecare staţie este mai mare la CDMA decât la FDM, iar problema alocării canalului este de asemenea rezolvată.

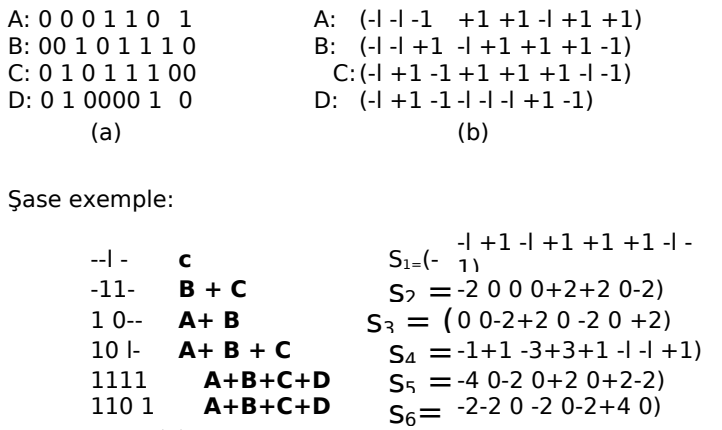


Fig.2-45. Secvenţele de felii binare pentru patru staţii. (b) Secvenţele de felii bipolare. (c) Şase exemple de transmisii. (d) Recuperarea semnalului staţiei C

Din motive pedagogice, este mai convenabilă folosirea unei notaţii bipolare, cu 0 binar reprezentat ca -l şi 1 binar reprezentat ca +1. Vom scrie secvenţele de felii în paranteze, deci un bit 1 pentru staţia A devine (-l -l -l +1 +1 -l +1 +1). În fig. 2-45(a) sunt prezentate secvenţele de felii binare asociate celor patru staţii luate ca exemplu. În fig. 2-45(b) ele sunt prezentate în notaţia noastră bipolară.

Fiecare staţie are propria sa secvenţă de felii. Vom folosi simbolul **S** pentru a indica vectorul de *m* felii pentru staţia S, şi S pentru negaţia sa. Toate secvenţele de felii sunt **ortogonale pe perechi**, prin aceasta înţelegând că produsul scalar normat al oricăror două secvenţe distincte de aşchii **S** şi **T** (notat ca **S*T**) este 0. Se cunoaşte cum se generează asemenea secvenţe de felii ortogonale, folosind metoda **codurilor Walsh**. În termeni matematici, ortogonalitatea secvenţelor de felii poate fi exprimată după cum urmează:

$$\mathbf{S} \cdot \mathbf{T} = \frac{1}{m} \sum_{i=1}^{m} S_i T_i = 0 \qquad (2\text{-}$$

Altfel spus, toate perechile care se pot forma sunt diferite între ele. Această proprietate de orto- gonalitate se va dovedi crucială mai târziu. De notat că dacă **S*T**=0 atunci şi **S** • **T** =0. Produsul scalar normat al oricărei secvenţe de felii cu ea însăşi este 1:

$$\mathbf{S} \cdot \mathbf{S} = \frac{1}{m} \sum_{i=1}^{m} S_i S_i = \frac{1}{m} \sum_{i=1}^{m} S_i^2 = \frac{1}{m} \sum_{i=1}^{m} (\pm 1)^2 = 1$$

Acest lucru se întâmplă deoarece fiecare din cei *m* termeni ai produsului scalar este 1 deci suma este *m*. De asemenea se observă că **S*** **S** = -1.

În timpul fiecărui interval de bit, o staţie poate să transmită un 1 emiţând propria secvenţă de felii, sau poate transmite un 0 emiţând complementul secvenţei sale de felii, sau poate să nu transmită nimic. Pentru moment, vom presupune că toate staţiile sunt sincronizate în timp, deci toate secvenţele de felii încep în acelaşi moment.

Când două sau mai multe staţii transmit simultan, semnalele lor bipolare se adună liniar. De exemplu, dacă într-un interval de felie trei staţii emit +1 şi una -l, rezultatul este +2. Putem privi aceasta ca pe o adunare de tensiuni: trei staţii emit +1 volt şi o staţie emite -1volt, rezultând 2 volţi.

În fig. 2-45(c) avem şase exemple de una sau mai multe staţii transmiţând simultan. În primul exemplu, C transmite un bit de 1, deci avem doar secvenţa de felii a lui C. În al doilea exemplu, atât B cât şi C transmit biţi de 1, deci vom obţine suma secvenţelor lor de felii bipolare, şi anume:

(-l -l +1 -l +1 +1 +1 -1) + (-l +1 -l +1 +1 +1 -l -1) = (-2 0 0 0 +2 +2 0 -2).

În cel de al treilea exemplu, staţia A emite un 1 iar staţia B emite un 0. Celelalte tac. În al patrulea exemplu, A şi C emit câte un bit 1, în timp ce B emite un 0. În al cincilea exemplu, toate cele 4 staţii emit câte un bit 1. În final, în ultimul exemplu, A, B şi D emit câte un bit 1, în timp ce C emite un bit 0. Să notăm că fiecare dintre

cele şase secvenţe, S_1 – S_6, date în fig. 2-45(c), reprezintă durata unui singur bit.

Pentru a reface şirul de biţi al unei staţii individuale, receptorul trebuie să cunoască dinainte secvenţa de felii a staţiei. El face recuperarea calculând produsul scalar normat între secvenţa de felii recepţionată (suma liniară a tuturor staţiilor care au transmis) şi secvenţa de felii a staţiei al cărei şir de biţi încearcă să-l refacă. Dacă secvenţa de felii recepţionată e **S** şi receptorul încearcă să asculte de la o staţie a cărei secvenţă de felii e **C**, atunci va calcula doar produsul scalar normat **S*C**.

Pentru a vedea de ce se întâmplă aşa, imaginaţi-vă două staţii A şi C, ambele transmiţând un bit 1 în acelaşi timp în care B transmite un bit 0. receptorul primeşte suma **S=A+ B +C** şi calculează:

$$\mathbf{S*C = (A + B + C) \bullet C = A*C + B*C + C*C = 0+0+1=1}$$

Primii doi termeni dispar, deoarece toate perechile de secvenţe de felii au fost alese cu grijă pentru a fi ortogonale, ca în ecuaţia (2-4). Acum ar trebui să fie clar de ce această proprietate trebuie impusă secvenţelor de felii.

Un alt mod de a gândi relativ la această situaţie este de a ne imagina toate cele trei secvenţe de felii sosite separat, în loc să fie însumate. Atunci receptorul ar calcula produsul scalar cu fiecare în parte şi ar aduna rezultatele. Datorită proprietăţii de ortogonalitate, toate produsele scalare în afară de **C*C** ar fi 0. Adunându-le şi apoi făcând produsul lor scalar, este de fapt acelaşi lucru cu calculul produselor scalare, şi adunarea acestora.

Pentru a concretiza procesul de decodificare, să considerăm din nou cele şase exemple din fig. 2- 45(c), ilustrat din nou în 2.45(d). Să presupunem că receptorul e interesat de extragerea bitului trimis de staţia C din fiecare din cele şase sume S_1 – S_6. Se calculează acest bit prin însumarea perechilor de produse între vectorul **S** recepţionat şi vectorul **C** din figura2-45(b), luând apoi 1/8 din rezultat (pentru că în acest caz m=8). Aşa cum am arătat, de fiecare dată este decodificat bitul corect. Este ca şi cum s-ar fi vorbit franceza.

Într-un sistem CDMA ideal, fără zgomote, capacitatea (adică numărul de staţii) poate fi mărită oricât de mult, aşa cum şi capacitatea unui canal Nyquist fără zgomote poate fi mărită oricât de mult, prin utilizarea unui număr tot mai mare de biţi pe eşantion. În practică, limitările fizice reduc considerabil capacitatea. La început, am presupus că toate feliile sunt sincronizate în timp. În realitate, aceasta este imposibil de realizat. Ceea ce se poate face este ca emiţătorul şi receptorul să se sincronizeze prin expedierea de către emiţător a unei secvenţe de felii cunoscute, suficient de lungă pentru ca receptorul să o poată localiza. Astfel, toate celelalte transmisii (nesincronizate) sunt percepute ca un zgomot aleator. Dacă ele nu sunt prea numeroase, algoritmul fundamental de decodificare funcţionează, totuşi, destul de bine. Există multă teorie referitoare la suprapunerea secvenţei de felii peste nivelul de zgomot (Pickholtz, şi alţii 1982). Aşa cum vă puteţi aştepta, cu cât secvenţa de felii este mai lungă, cu atât este mai mare probabilitatea detectării ei corecte în prezenţa zgomotului. Pentru o mai mare siguranţă, secvenţa de biţi poate folosi un cod corector de erori. Secvenţele de felii nu folosesc niciodată

coduri corectoare de erori.

O presupunere implicită în discuţia anterioară este că nivelurile de putere ale tuturor staţiilor sunt aceleaşi cu cele percepute de receptor. Protocolul CDMA este folosit în mod obişnuit pentru sistemele fără fir cu o staţie de bază fixă şi multe staţii mobile la distanţe variabile de aceasta. Nivelurile de putere, recepţionate la staţia de bază depind de cât de departe sunt emiţătorii. În acest caz, o euristică bună este ca fiecare staţie mobilă să emită către staţia de bază la un nivel complementar de putere faţă de cel primit de la staţia de bază. Altfel spus, o staţie mobilă care recepţionează un semnal slab va folosi mai multă putere decât una care primeşte un semnal puternic. De asemenea, staţia de bază dă comenzi explicite către staţiile mobile pentru a-şi creşte/descreşte puterea de transmisie.

De asemenea, am presupus că receptorul ştie cine este emiţătorul. În principiu, dată fiind suficientă putere de calcul, receptorul poate asculta toţi emiţătorii în acelaşi timp, prin rularea algoritmului de decodificare pentru fiecare dintre ei în paralel. În viaţa reală, este de ajuns să spunem că e mai uşor de zis decât de făcut. De asemenea, CDMA are mulţi alţi factori care complică soluţia şi care au fost comentaţi în această scurtă introducere. Totuşi, CDMA este o schemă inteligentă care este rapid introdusă pentru comunicaţii mobile fără fir. În mod normal, operează într-o bandă de 1.25 MHz (faţă de 30 KHz pentru D-AMPS şi 200 KHz pentru GSM), dar suportă în aceeaşi bandă mai mulţi utilizatori decât oricare dintre celelalte sisteme. În practică, lărgimea de bandă disponibilă pentru fiecare utilizator este cel puţin la fel de bună ca la GSM şi deseori mult mai bună.

Inginerii care doresc să înţeleagă foarte bine CDMA ar trebui să citească (Lee şi Miller, 1998). O schemă alternativă de împrăştiere, în care împrăştierea se face în timp, şi nu în frecvenţă, este descrisă în (Crespo et al., 1995). Încă o schemă este descrisă în (Sari et al., 2000). Toate aceste referinţe necesită ceva cunoştinţe în ingineria comunicaţiilor.

2.6.3 A treia generaţie de telefoane mobile: Voce digitală şi date

Care este viitorul în telefonia mobilă? Să aruncăm o privire rapidă. Un număr de factori dirijează industria. În primul rând, traficul de date depăşeşte deja traficul de voce pe reţeaua fixă şi creşte exponenţial, pe când traficul de voce este constant. Mulţi experţi din industrie se aşteaptă ca traficul de date să domine traficul de voce şi pe dispozitivele mobile, cât de curând. În al doilea rând, industriile telefonice, de divertisment şi de calculatoare au devenit toate digitale şi converg rapid. Mulţi oameni îşi doresc un dispozitiv uşor, portabil care să se comporte ca un telefon, CD player, DVD player, terminal pentru poşta electronică, interfaţă Web, staţie pentru jocuri, procesor de text, şi chiar mai mult, totul însoţit de conectivitate internaţională fără fir la Internet cu lărgime de bandă ridicată. Dispozitivul şi modul de conectare al acestuia sunt ceea ce înseamnă generaţia a treia de telefonie mobilă. Pentru mai multe informaţii, vezi (Huber et al., 2000; şi Sarikaya, 2000).

În 1992, ITU a încercat să fie mai precis în ceea ce priveşte acest vis şi a iniţiat un plan pentru a-şi atinge scopul, numit **IMT-2000,** unde IMT înseamnă **International Mobile Telecommunications** (rom: Telecomunicaţii Mobile

Internaţionale). Numărul 2000 vine de la trei lucruri: (1) anul în care trebuia să intre în utilizare, (2) frecvenţa la care trebuia să funcţioneze (în MHz), şi (3) lărgimea de bandă pe care serviciul trebuia să o aibă (în KHz).

Nu a reuşit în nici unul dintre aspectele propuse. Nimic nu a fost implementat până în 2000. ITU a recomandat ca toate guvernele să rezerve spectru la 2 GHz, astfel încât dispozitivele să comunice fără redirectări între ţări. China a rezervat respectiva bandă de frecvenţă, dar nimeni altcineva nu mai făcut la fel. În sfârşit, a fost recunoscut că 2 Mbps nu este în prezent posibil pentru utilizatori care sunt prea mobili (datorită dificultăţii de a realiza schimburile suficient de repede). Mai realist este 2 Mbps pentru utilizatori staţionari, în spaţii închise (ceea ce va rivaliza direct cu ADSL), 384 Kbps pentru oameni care merg, şi 144 Kbps pentru conexiuni din maşini. Oricum, întreaga arie a **3G**, după cum este denumită, este un larg domeniu de activitate. A treia generaţie se poate să fie un pic mai puţin decât se aştepta şi un pic mai târzie, dar este sigur că va apărea.

Principalele servicii pe care reţeaua IMT-2000 ar trebui să le ofere utilizatorilor săi sunt:

1. Transmisie de voce de înaltă calitate.
2. Mesagerie (înlocuirea poştei electronice, fax-ului, SMS-ului, chat-ului etc.).
3. Multimedia (muzică, vizualizare video, filme, televiziune) .
4. Acces la Internet (navigare pe Web, inclusiv pagini cu audio şi video).

Servicii adiţionale ar putea fi videoconferinţele, prezentările televizate, jocurile în grup şi comerţul mobil (fluturarea mobilului către casieră pentru a plăti într-un magazin). Mai mult, toate aceste servicii ar trebui să fie disponibile oriunde pe glob (prin conectare automată printr-un satelit atunci când nici o reţea terestră nu poate fi localizată), instant (întotdeauna conectat), şi cu garanţii pentru calitatea serviciului.

ITU a imaginat o singură tehnologie internaţională pentru IMT-2000, astfel încât fabricanţii să poată construi un singur dispozitiv care să fie vândut şi folosit oriunde în lume (cum ar fi CD player- ele şi calculatoarele şi nu cum sunt telefoanele mobile şi televizoarele). O singură tehnologie ar face viaţa mult mai simplă şi pentru operatorii de reţea şi ar încuraja mai mulţi oameni să folosească serviciile. Războaiele de format, cum a fost cel dintre Betamax şi VHS atunci când au apărut videocase- tofoanele, nu sunt benefice pentru afaceri.

Au fost făcute câteva propuneri, iar după mai multe trieri au rămas în discuţie două propuneri mai interesante. Prima, **W-CDMA,** adică **Wideband CDMA** (rom: CDMA de bandă largă), a fost propusă de Ericsson. Acest sistem foloseşte secvenţă directă în spectru larg de tipul descris mai sus. Funcţionează la o lărgime de bandă de 5 MHz şi a fost proiectat să interacţioneze cu reţele GSM actuale, fără a încerca să includă şi compatibilitatea cu versiuni GSM anterioare. Are, totuşi, proprietatea că un apelant poate părăsi o celulă W-CDMA şi poate intra într-o celulă GSM fără a pierde apelul. Acest sistem a fost puternic susţinut de Uniunea Europeană, care l-a numit **UMTS** (**Universal Mobile Telecommunications System**, rom: sistem universal de telecomunicaţii mobile) .

Cealaltă propunere a fost **CDMA2000**, propus de Qualcomm. Şi acesta este un proiect bazat pe secvenţa directă în spectru larg, fiind de fapt o extensie a lui IS-95 şi compatibil cu acesta. Foloseşte de asemenea o bandă de frecvenţă de 5 MHz, dar nu a fost proiectat să interacţioneze cu GSM şi nu poate redirecta apeluri către o celulă GSM (sau o celulă D-AMPS). Alte diferenţe tehnice faţă de W-CDMA sunt vitezele de cip diferite, timpii de cadre diferiţi, spectru diferite şi moduri diferite de sincronizare.

Dacă inginerii de la Ericsson şi de la Qualcomm ar fi fost închişi într-o singură cameră şi li s-ar fi spus să găsească o soluţie comună, probabil că ar fi reuşit. De fapt, principiul de bază din spatele ambelor sisteme este CDMA într-un canal de 5 MHz şi nimeni nu vrea să moară pentru viteza de cip preferată. Necazul este că problema reală nu este ingineria, ci politica (ca de obicei). Europa a vrut un sistem care să poată interacţiona cu GSM, iar S.U.A. doreau un sistem care să fie compatibil cu unul deja popular în Statele Unite (IS-95). În plus, fiecare parte a sprijinit compania locală (Ericsson este originară din Suedia, iar Qualcomm este din California). În fine, Ericsson şi Qualcomm au fost implicate în numeroase procese privind patentele asupra respectivelor proiecte CDMA.

În martie 1999, cele două companii au terminat cu procesele atunci când Ericsson a fost de acord să cumpere infrastructura Qualcomm. S-au înţeles de asemenea asupra unui singur standard 3G, dar unul cu multe opţiuni incompatibile, care până la urmă recunoaşte pe hârtie diferenţele tehnice. Aceste dispute fiind încheiate, dispozitivele 3G şi serviciile ar putea începe să apară în viitorii ani.

Multe s-au scris despre sistemele 3G, multe lucrări lăudându-le ca pe cel mai mare lucru de la pâinea feliată încoace. Unele referinţe sunt (Collins şi Smith, 2001; De Vriendt et al., 2002; Harte et al., 2002; Lu, 2002 şi Sarikaya, 2000). Totuşi, unii pesimişti încă mai cred că industria se îndreaptă în direcţia greşită (Garber, 2002 şi Goodman, 2000).

În aşteptarea sfârşitului disputei asupra 3G, unii operatori fac cu prudenţă un pas mic şi atent în direcţia 3G prin a merge spre ceea ce este numit uneori **2.5G**, deşi 2.1G este mai exact. Un astfel de sistem este **EDGE** (**Enhanced Data rates for GSM Evolution**, rom: viteze de transfer de date mărite pentru evoluţia GSM), ceea ce este chiar GSM cu mai mulţi biţi pe baud. Problema este că mai mulţi biţi pe baud înseamnă mai multe erori pe baud, astfel că EDGE dispune de nouă scheme diferite de modulare şi corectare de erori, în funcţie de ce parte din bandă este dedicat corectării erorilor introduse de viteza mai mare.

O altă schemă 2.5G este **GPRS** (**General Packet Radio Service**, rom: serviciul radio pentru pachete generice), care este o reţea de pachete generale peste D-AMPS sau GSM. Ea permite staţiilor
mobile să trimită şi să primească pachete IP într-o celulă folosind sistemul pentru voce. În timpul operării GPRS, unele intervale de timp ale unor frecvenţe sunt rezervate pentru traficul de pachete. Numărul şi poziţia intervalelor pot fi administrate dinamic de către staţia de bază, în funcţie de raportul între de trafic de voce şi de date din celulă.

Intervalele de timp disponibile sunt împărţite în câteva canale logice, folosite în diverse scopuri. Staţia de bază determină alocarea de canale logice peste

intervalele de timp. Un canal logic este utilizat pentru preluarea de pachete de la staţia de bază la o staţie mobilă, fiecare pachet indicând cui îi este destinat. Pentru a trimite un pachet IP, o staţie mobilă solicită unul sau mai multe intervale de timp prin trimiterea unei cereri la staţia de bază. Dacă această cerere ajunge fără probleme, staţia de bază anunţă frecvenţa şi intervalele de timp alocate mobilului pentru a trimite pachetul. O dată ce pachetul a ajuns la staţia de bază, este transferat în Internet printr-o conexiune cu fir. Deoarece GPRS este doar un nivel suplimentar suprapus peste sistemul de voce deja existent, el reprezintă în cel mai bun caz o soluţie de umplere a golului până la sosirea lui 3G.

Deşi reţelele 3G nu sunt complet dezvoltate deocamdată, unii cercetători privesc 3G ca pe o afacere încheiată şi, deci, care nu mai prezintă interes. Aceşti oameni lucrează deja la sistemele 4G (Berezdivin et al., 2002; Guo şi Chaskar, 2002; Huang şi Zhuang, 2002; Keller et al., 2002; şi Misra et al., 2002). Unele dintre caracteristicile propuse pentru sistemele 4G includ lărgime de bandă mare, omniprezenţă (conectivitate oriunde), integrare simplă cu reţelele cablate şi mai ales cu IP, administrare de resurse şi spectru, transmisii radio software şi calitatea serviciilor pentru aplicaţii multimedia.

Pe de altă parte, sunt amenajate la tot pasul o sumedenie de puncte de acces pentru LAN-urile fără fir 802.11, astfel încât unii oameni cred că 3G este nu numai o afacere neîncheiată, ci şi că este condamnat la dispariţie. În această viziune, oamenii se vor plimba de la un punct de acces 802.11 la altul pentru a rămâne conectaţi. A spune că „industria este într-o continuă creştere” este doar o imensă subestimare. Reveniţi la această discuţie peste 5 ani, ca să vedeţi ce s-a întâmplat.

2.7 TELEVIZIUNEA PRIN CABLU

Am studiat până acum atât sistemele de telefonie fixă cât şi pe cele de telefonie fără fir în mod egal. Ambele vor juca un rol important în reţelele viitorului. Totuşi, o nouă alternativă disponibilă pentru reţelele fixe devine acum un jucător important: reţelele de televiziune prin cablu. Mulţi oameni primesc deja telefonul şi serviciile Internet prin cablu, iar operatorii de cablu muncesc din greu pentru a-şi creşte cota pe piaţă. În următoarele paragrafe vom privi mai în detaliu televiziunea prin cablu ca sistem de reţele şi îl vom compara cu sistemele de telefonie pe care tocmai le-am studiat. Pentru mai multe informaţii despre televiziunea prin cablu, vezi (Laubach et al.,2001; Louis, 2002; Ovadia,2001; şi Smith, 2002).

2.7.1 Televiziune prin antena colectivă

Televiziunea prin cablu fost concepută la sfârşitul anilor 1940 ca un mijloc de a oferi recepţie mai bună oamenilor care trăiau în zone rurale sau muntoase. Iniţial, sistemul era format dintr-o antenă mare montată pe vârful unui deal pentru a capta semnalul de televiziune din aer, un amplificator, numit **amplificator terminal (**eng: **head end)**, pentru a spori puterea semnalului, şi un cablu coaxial pentru a

transmite semnalul în casele oamenilor, după cum este ilustrat în fig. 2-46.

În primii ani, televiziunea prin cablu era numită t**eleviziune prin antena colectivă**. Se asemăna foarte mult cu o operaţie mama-şi-tata; oricine era destul de îndemânatic la electronică putea pune la punct un serviciu pentru oraşul lui, pentru ca apoi utilizatorii să participe la plata costurilor. Deoarece numărul de abonaţi a crescut, au trebuit adăugate cabluri în derivaţie din cablul original şi s-au adăugat noi amplificatoare acolo unde a fost nevoie de ele. Transmisia era într-un singur sens, de la amplificatorul din capăt, de lângă antenă, spre utilizatori. Până în 1970, existau mii de astfel de sisteme independente.

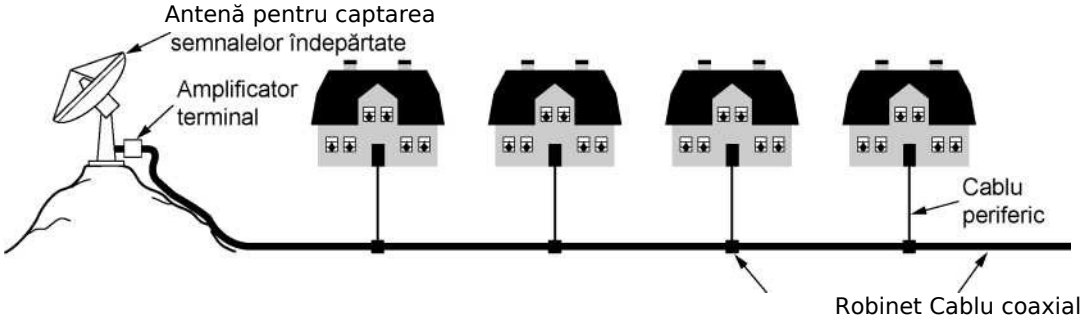


Fig. 2-46. Un sistem primitiv de televiziune prin cablu

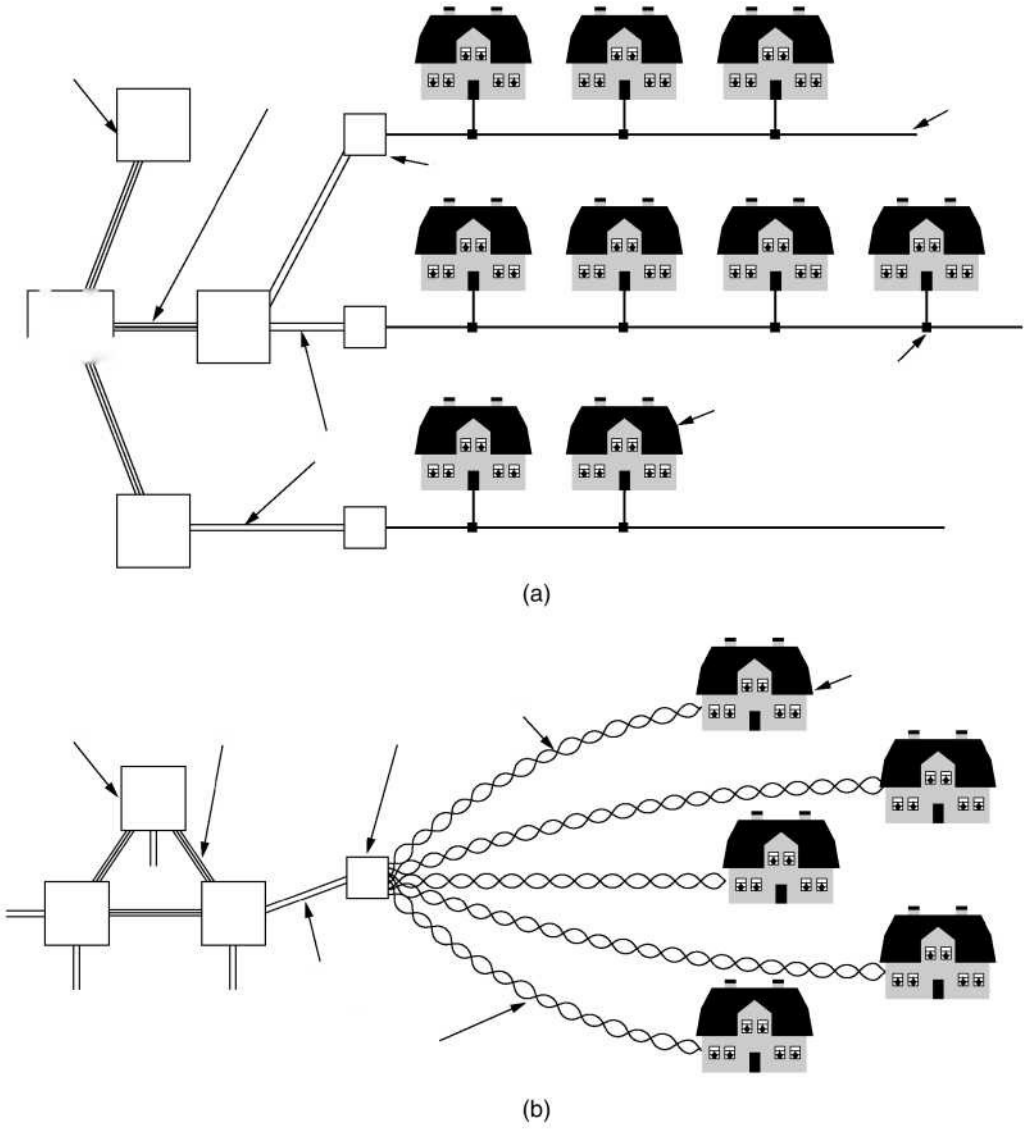
În 1974, Time Inc., a pus bazele unui nou canal, Home Box Office, cu conţinut nou (filme) şi distribuit doar prin cablu. Au urmat alte canale distribuite doar pe cablu, cu ştiri, sport, gastronomie şi multe alte subiecte. Această dezvoltare a dat naştere la două schimbări în industrie. În primul rând, mari corporaţii au început să cumpere sisteme de cablu deja existente şi să întindă noi cabluri pentru a atrage noi abonaţi. În al doilea rând, exista acum cerinţa de a se conecta mai multe sisteme de cablu, adesea aflate în oraşe îndepărtate, pentru a fi distribuite canalele noi de televiziune prin cablu. Companiile de cablu au început să întindă cabluri între oraşele lor pentru a le conecta pe toate într-un sistem unic. Acest model a fost analog cu ceea ce s-a întâmplat în industria telefonică în urmă cu 80 de ani în cazul conectării oficiilor periferice izolate pentru a face posibile apelurile la distanţă mare.

2.7.2 Internet prin cablu

De-a lungul anilor sistemul de cablu a crescut şi cablurile dintre diversele oraşe au fost înlocuite cu fibră cu lărgime de bandă mare, similar cu ceea ce s-a întâmplat în sistemul telefonic. Un sistem cu fibră pentru transportul pe distanţe lungi şi cu cablu coaxial până la casele clienţilor este numit sistem **HFC** (**Hybrid Fiber Coax**, rom: hibrid fibră-coaxial). Convertoarele electro-optice folosite pentru interfaţarea părţilor optice şi electrice ale sistemului sunt numite **noduri de fibră**. Pentru că lărgimea de bandă a fibrei este mult mai mare decât cea a cablului coaxial, un nod de fibră poate alimenta mai multe cabluri coaxiale. O parte a unui sistem HFC modern este prezentat în fig. 2-47(a).

În ultimii anii, mulţi operatori de cablu au decis să intre în afaceri legate de

accesul la Internet, şi adesea şi în afaceri legate de telefonie. Totuşi, diferenţele tehnice dintre centralele de cablu şi cele de telefon au efecte semnificative asupra operaţiilor care trebuie făcute pentru a atinge aceste scopuri. De exemplu, toate amplificatoarele unidirecţionale din sistem trebuie înlocuite cu amplificatoare bidirecţionale.



Totuşi, există între sistemul HFC din fig. 2-47(a) şi sistemul telefonic din fig. 2-47(b) o diferenţă care este mult mai greu de înlăturat. În inima cartierelor, un singur cablu este utilizat în comun de mai multe case, pe când în sistemul telefonic, fiecare casă are propria buclă locală. Când este folosit pentru difuzarea televiziunii, această utilizare multiplă nu joacă nici un rol. Toate programele sunt difuzate pe cablu şi nu are nici o importanţă dacă sunt 10 telespectatori sau 10.000 de telespectatori. Când acelaşi cablu este utilizat pentru acces la Internet, contează foarte mult dacă sunt 10 utilizatori sau 10.000. Dacă unul dintre utilizatori se

Comutato
Trunchi de fibră de
lărgime de
Cablu coaxial
Nod de
Ampli-ficator termina
Robine

decide să preia de pe Internet un fişier foarte mare, lărgimea de bandă respectivă este teoretic luată de la alţi utilizatori. Cu cât sunt mai mulţi utilizatori, cu atât este mai mare competiţia pentru lărgimea de bandă. Sistemul telefonic nu are această caracte

ristică: salvarea unui fişier foarte mare prin intermediul unei linii ADSL nu reduce banda de frecvenţă a vecinului. Pe de altă parte, lărgimea de bandă a cablului coaxial este mult mai mare decât cea a perechilor torsadate.

Soluţia prin care industria cablului a rezolvat această problemă a fost separarea cablurilor lungi şi conectarea fiecăreia dintre bucăţi direct la un nod de fibră. Lărgimea de bandă de la amplificator pân ă la fiecare nod de fibră este teoretic infinită, aş a că atâta timp cât nu sunt foarte mulţi abonaţi pe fiecare segment de cablu, cantitatea de trafic este rezonabilă. În prezent, cablurile obişnuite deservesc 500-2000 de case. Din ce în ce mai mulţi oameni se abonează la Internet prin cablu, aşa că încărcarea ar putea să devină prea mare, necesitând mai multe separări şi mai multe noduri de fibră.

2.7.3 Alocarea de spectru

Renunţarea la toate canalele TV şi utilizarea infrastructurii de cablu strict pentru accesul la Internet ar genera probabil un număr considerabil de clienţi iritaţi, astfel încât companiile de cablu ezită să facă acest lucru. Mai mult, cele mai multe oraşe cenzurează serios ceea ce este pe cablu, astfel că operatorii de cablu nu ar putea face acest lucru nici dacă ar dori într-adevăr. Ca o consecinţă, a fost nevoie să se găsească o modalitate ca televiziunea şi Internetul sa coexiste pe acelaşi cablu.

Canalele televiziunii prin cablu din America de Nord ocupă în mod normal regiunea de 54-550 MHz (mai puţin regiunea de la 88 la 108 MHz, atribuită radioului FM). Aceste canale au 6 MHz lăţime, inclusiv benzile de siguranţă. În Europa, limita inferioară este de obicei de 65 MHz şi canalele au 6-8 MHz lăţime pentru rezoluţia mai înaltă cerută de PAL şi SECAM, dar altfel schema de alocare este similară. Partea de jos a benzii nu este folosită. De asemenea, cablurile moderne pot opera cu mult peste 550 MHz, adesea la 750 MHz sau mai mult. Soluţia aleasă a fost introducerea de canale ascendente în banda 5-42 MHz (ceva mai sus în Europa) şi utilizarea frecvenţelor de la limita superioară pentru canale descendente. Spectrul cablului este ilustrat în fig. 2-48.

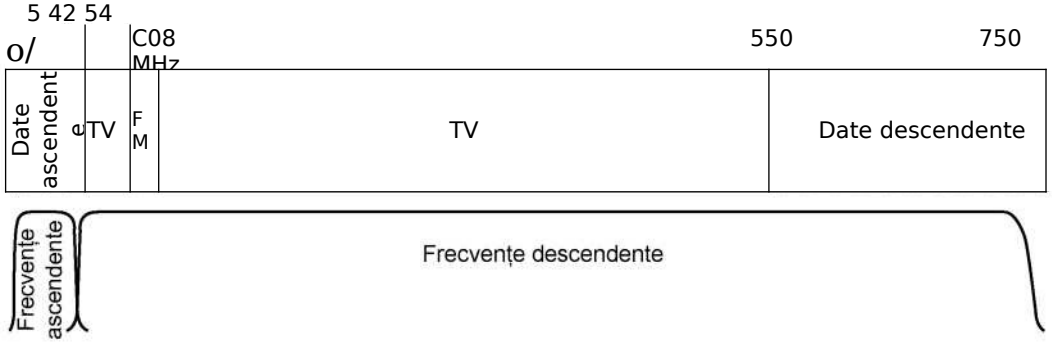


Fig. 2-48. Alocarea frecvenţelor într-un sistem tipic de TV prin cablu folosit pentru

Observaţi că din momentul în care semnalele de televiziune sunt toate

descendente, este posibil să se folosească amplificatoare ascendente care funcţionează doar în regiunea 5-42 MHz şi amplificatoare descendente care funcţionează doar la 54 MHz şi mai sus, după cum se vede în figură. Astfel, obţinem o asimetrie între benzile ascendente şi descendente, pentru că este disponibil mai mult spectru deasupra spectrului televiziunii, decât sub acesta. Pe de altă parte, mare parte din trafic este probabil descendent, astfel că operatorii de cablu nu sunt supăraţi din această cauză. După cum am văzut mai devreme, şi companiile telefonice oferă de obicei un serviciu DSL asimetric, chiar dacă nu au nici un motiv tehnic să facă acest lucru.

Cablurile coaxiale lungi nu sunt cu nimic mai bune în transmiterea semnalelor digitale decât sunt buclele locale lungi, astfel că modularea analogică este necesară şi aici. Schema uzuală este să se ia fiecare canal descendent de 6 MHz sau 8 MHz şi să se moduleze cu QAM-64 sau, pentru o calitate excepţională a transmisiei prin cablu, QAM-256. Cu un canal de 6 MHz şi QAM-64 obţinem 36 Mbps. După ce scădem supraîncărcarea, informaţia utilă netă este de aproape 27 Mbps. Cu QAM-256, informaţia utilă netă este aproape 39 Mbps. Valorile europene sunt cu 1/3 mai mari.

Pentru canalele ascendente, chiar şi QAM-64 nu funcţionează prea bine. Este prea mult zgomot de la microundele terestre, radiourile CB şi alte surse, astfel că se foloseşte o schemă mai conservativă – QPSK. Această metodă (prezentată în fig. 2-25) generează 2 biţi pe baud în loc de 6 sau 8 biţi pe care QAM îi furnizează pe canalele descendente. În consecinţă, asimetria dintre banda descendentă şi cea ascendentă este mai profundă decât sugerează fig. 2-48.

Pe lângă actualizarea amplificatoarelor, operatorul trebuie să actualizeze şi capătul de pornire al cablului, transformându-l dintr-un amplificator primitiv într-un sistem computerizat digital inteligent cu o interfaţă de lărgime de bandă mare între fibră şi ISP. Adesea şi numele este actualizat, din amplificator terminal al cablului în **CTMS** (**Cable Modem Termination System**, rom: sistem terminal pentru modemuri de cablu). În continuarea acestui text, vom evita să facem o actualizare de nume şi vom menţine tradiţionalul amplificator terminal al cablului (headend).

2.7.4 Modemuri de cablu

Accesul la Internet prin cablu necesită un modem pentru cablu, dispozitiv care conţine două interfeţe: una către calculator şi una către reţeaua de cablu. În primii ani ai Internetului prin cablu, fiecare operator avea un modem propriu patentat pentru cablu, care era instalat de un tehnician al companiei de cablu. Totuşi, a devenit în curând evident că un standard ar crea o piaţă competitivă de modemuri pentru cablu şi ar scădea preţurile, încurajând astfel utilizarea serviciului. Mai mult, clienţii care cumpără modemurile pentru cablu din magazin şi le instalează singuri (la fel cum procedează cu modemurile telefonice V.9x) ar putea scuti firma de cheltuielile intervenţiilor la domiciliu.

În consecinţă, operatorii de cablu mai importanţi au făcut echipă cu o companie numită CableLabs pentru a produce un standard de modem de cablu şi pentru a testa produsele din punct de vedere al compatibilităţii. Acest standard, numit

DOCSIS (**Data Over Cable Service Interface Specification**, rom: specificaţiile interfeţei serviciului de date prin cablu) abia începe să înlocuiască modemurile patentate. Versiunea europeană se numeşte **EuroDOCSIS**. Nu tuturor operatorilor de cablu le place ideea unui standard, deoarece mulţi dintre ei au făcut bani frumoşi prin închirierea modemurilor lor către clienţii lor captivi. Un standard public cu zeci de fabricanţi care vând modemuri de cablu în magazine ar pune punct acestei practici profitabile.

Interfaţa modem-calculator este directă. Uzual, este 10 Mbps Ethernet (sau ocazional USB) în prezent. În viitor, întregul modem ar putea fi un card mic, conectat direct la calculator, la fel cum sunt modemurile interne V.9x .

Celălalt capăt este mai complicat. O mare parte a standardului se referă la ingineria radio, un subiect care este cu mult în afara scopului acestei cărţi. Singura parte care merită să fie menţionată aici este aceea că modemurile pentru cablu, ca şi modemurile ADSL, sunt întotdeauna conectate. Ele deschid o conexiune când sunt activate şi menţin acea conexiune atât timp cât sunt alimentate pentru că operatorii de cablu nu percep taxe diferite în funcţie de durata conexiunii.

Pentru a înţelege mai bine cum funcţionează aceste modemuri, să vedem ce se întâmplă atunci când un modem de cablu este conectat la reţeaua electrică şi pornit. Modemul scanează canalele descendente căutând un pachet special, este trimis periodic de către alimentatorul terminal (headend) pentru a furniza parametrii sistemului către modemurile care tocmai s-au conectat. După ce detectează acest pachet, noul modem îşi anunţă prezenţa pe unul dintre canalele ascendente. Amplificatorul terminal răspunde alocându-i modemului canalele corespunzătoare pentru flux ascendent şi descendent. Aceste alocări pot fi schimbate mai târziu dacă amplificatorul terminal considera că este necesar să echilibreze încărcarea.

Modemul determină apoi distanţa de la amplificatorul terminal trimiţând către acesta un pachet special şi măsurând cât îi ia să primească răspunsul. Acest proces se numeşte **ranging** (rom: poziţionare). Este important pentru modem să cunoască aceasta distanţă pentru a regla modul de funcţionare a canalelor de flux ascendent şi pentru a reuşi sincronizarea. Aceste canale sunt divizate în timp în **mini-intervale** (eng. minislots). Fiecare pachet de flux ascendent trebuie să încapă într-unul sau mai multe mini-intervale consecutive. Amplificatorul terminal anunţă periodic începutul unei noi serii de mini-intervale, dar pistolul de start nu se aude simultan la toate modemurile, din cauza timpului de propagare prin cablu. Cunoscând cât de departe se află de capăt, fiecare modem poate să calculeze cu cât timp în urmă a început cu adevărat primul mini-interval. Lungimea mini-intervalelor este dependentă de reţea. O încărcare tipică este de 8 octeţi.

În timpul iniţializării, amplificatorul terminal alocă fiecărui modem şi câte un mini-interval pentru a cere lăţime de bandă de flux ascendent. Ca o regulă, mai multor modemuri li se va aloca acelaşi mini-interval, ceea ce duce la conflicte. Atunci când un calculator vrea să transmită un pachet, el transferă pachetul modemului, care cere apoi numărul de mini-intervale necesare pentru el. Dacă această cerere este acceptată, amplificatorul terminal transmite o confirmare pe

canalul de flux descendent, spunându-i modemului ce mini-intervale au fost rezervate pentru pachetul său. Pachetul este apoi trimis, începând de la mini-intervalul alocat lui. Pentru a solicita transmisia de pachete adiţionale se foloseşte un câmp din antet.

Pe de altă parte, dacă apare un conflict pentru mini-intervalul cerut, nu va exista nici o confirmare, iar modemul va aştepta un timp oarecare şi apoi va încerca din nou. După fiecare eşec succesiv, timpul aleator maxim se dublează. (Pentru cititorii deja familiarizaţi într-o oarecare măsură cu reţelele, acest algoritm este ALOHA discretizat cu regresie exponenţială binară. Nu se poate folosi Ethernet pe cablu pentru că staţiile nu pot asculta mediul. Vom reveni la aceste teme în Cap.4).

Canalele descendente sunt administrate diferit faţă de cele ascendente. În primul rând, avem un singur emiţător (amplificatorul terminal), deci nu apar conflicte şi nu este nevoie de mini-intervale, care reprezintă de fapt o multiplexare statistică prin divizarea timpului. În al doilea rând, traficul descendent este de obicei mult mai mare decât cel ascendent, astfel încât se foloseşte o dimensiune de pachet fixată la 204 octeţi. O parte din acest pachet reprezintă codul corector de erori Reed- Solomon şi alte informaţii suplimentare, lăsând utilizatorului un pachet util de 184 octeţi. Aceste numere au fost alese pentru compatibilitate cu televiziunea digitală care foloseşte MPEG-2, astfel încât canalele TV şi canelele de flux descendent de date să fie formatate identic. La nivel logic, conexiunile sunt prezentate în fig. 2-49.

Revenind la iniţializarea modemului, o data ce un modem a terminat poziţionarea şi şi-a căpătat canalul de flux ascendent, canalul de flux descendent şi alocarea mini-intervalelor, este liber să înceapă să trimită pachete. Primul pachet este trimis către ISP cerând o adresă IP, care este alocată dinamic folosind un protocol numit DHCP, pe care îl vom studia în cap. 5. De asemenea cere şi primeşte ora exactă de la amplificatorul terminal.

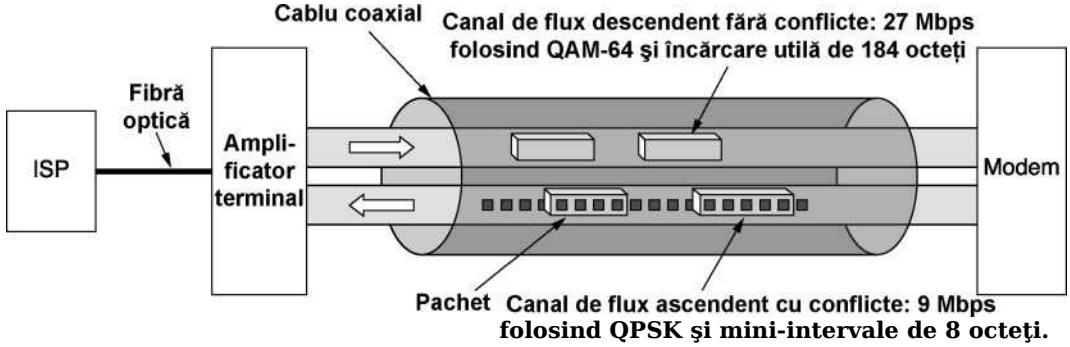

Cablu coaxial
Canal de flux descendent fără conflicte: 27 Mbps
folosind QAM-64 şi încărcare utilă de 184 octeți
Fibră optică
ISP
Ampli-ficator terminal
Modem
Pachet
Canal de flux ascendent cu conflicte: 9 Mbps
folosind QPSK şi mini-intervale de 8 octeţi.

Fig. 2-49. Detaliile tipice pentru canalele de flux de date ascendent şi descendent în America de Nord

Următorul pas implică securitatea. Cablul fiind un mediu partajat, oricine îşi dă silinţa poate să citească tot traficul care trece pe lângă el. Pentru a preveni situaţia în care oricine poate să îşi asculte vecinii (la propriu), tot traficul este criptat în ambele direcţii. O parte a procedurii de iniţializare presupune stabilirea cheilor de criptare. La început cineva ar putea să creadă că a avea doi străini, amplificatorul

terminal şi modemul care să stabilească o cheie secreta în plină zi cu mii de oameni uitându-se la ei ar fi imposibil. Se va demonstra că nu e aşa, dar trebuie să aşteptăm până în Cap. 8 ca să explicam cum (răspunsul, pe scurt: folosind algoritmul Diffie-Hellman).

În sfârşit, modemul trebuie să se conecteze şi să ofere identificatorul său unic prin canalul sigur. În acest moment, iniţializarea este completă. Utilizatorul poate să se conecteze la ISP şi să treacă la treabă.

Se mai pot spune multe lucruri despre modemurile de cablu. Unele referinţe relevante sunt (Adams şi Dulchinos, 2001; Donaldson şi Jones, 2001 şi Dutta-Roy, 2001).

2.7.5 Comparaţie între ADSL şi cablu

Care variantă este mai bun ă, ADSL sau cablul? Este ca şi cum ai întreba care sistem de operare este mai bun. Sau care limbă este mai bună. Sau care religie. Răspunsul primit depinde de persoana întrebată. Să comparăm ADSL şi cablul din câteva puncte de vedere. Ambele folosesc fibra optică drept coloană vertebrală, dar diferă la periferie. Sistemele prin cablu folosesc coaxial; ADSL foloseşte perechi torsadate. Capacitatea teoretică de transport a cablului este de sute de ori mai mare decât cea a perechii torsadate. Totuşi, nu este disponibilă întreaga capacitate a cablului pentru utilizatorii de date, pentru că mare parte din lăţimea de bandă este irosită de chestiuni nefolositoare cum sunt programele de televiziune.

Practic, este greu să facem o estimare generală pentru capacitatea efectivă. Furnizorii de ADSL fac anumite declaraţii despre lăţimea de bandă (de exemplu, 1 Mbps flux descendent, 256 Kbps flux ascendent) şi în general reuşesc să onoreze în mod consistent cam 80 % din acestea. Furnizorii de cablu nu pot să declare nimic, deoarece capacitatea efectivă depinde de câţi utilizatori sunt activi în acel moment pe segmentul de cablu al utilizatorului. Uneori poate fi mai bine decât cu ADSL, alteori poate fi mai rău. Ceea ce ar putea fi deranjant, totuşi, este tocmai această variaţie imprevizibilă a calităţii serviciilor. Faptul că ai servicii excelente acum nu garantează servicii excelente peste un minut, deoarece se poate ca marele devorator de lărgime de bandă din oraş să-şi fi pornit calculatorul.

Chiar dacă un sistem ADSL câştigă mai mulţi utilizatori, creşterea numărului lor nu are un efect semnificativ asupra utilizatorilor existenţi, pentru că fiecare utilizator are o conexiune dedicată. În cazul cablului, pe măsură ce mai mulţi clienţi se abonează la servicii Internet, performanţa pentru utilizatorii existenţi va scădea. Singurul remediu pentru operatorul de cablu este să separe cablurile aglomerate şi să conecteze fiecare dintre ele direct la un nod de fibra optică. Acest lucru costă timp şi bani, deci există presiuni financiare pentru a-l evita.

Ca o paranteză, am studiat deja un alt sistem bazat, la fel ca şi cablul, pe folosirea unui canal partajat: sistemul telefonic mobil. Şi aici, un grup de utilizatori, pe care i-am putea numi colegi de celula, împart o cantitate fixă de lărgime de bandă. În mod normal, aceasta este divizată prin FDM şi TDM în felii fixe, alocate pentru utilizatorii activi, traficul de voce fiind destul de uniform. Dar pentru traficul

de date, aceasta împărţire rigidă este foarte ineficientă, deoarece utilizatorii de date sunt deseori în aşteptare, caz în care banda rezervată lor este irosită. Totuşi, din acest punct de vedere, accesul prin cablu se aseamănă mai mult cu sistemul de telefonie mobilă decât cu acela de telefonie fixă.

Disponibilitatea serviciului este un aspect în care ADSL şi cablul diferă. Toată lumea are un telefon, dar nu toţi utilizatorii sunt destul de aproape de oficiul final pentru a obţine ADSL. Pe de altă parte, nu toata lumea are televiziune prin cablu, dar dacă ai cablu şi compania ta de cablu oferă acces Internet, poţi să te abonezi. Distanţa până la nodul de fibră optică sau până la amplificatorul terminal nu reprezintă o problemă. De asemenea, merită menţionat că, deoarece cablul a pornit ca un mediu de distribuţie al televiziunii, puţine companii folosesc aşa ceva.

Fiind un mediu punct-la-punct, ADSL este prin definiţie mai sigur decât cablul. Orice utilizator de cablu poate citi cu uşurinţă pachetele ce vin pe cablu. Din acest motiv, orice furnizor de cablu decent va cripta traficul în ambele direcţii. Oricum, chiar dacă ştii că acela care îţi citeşte mesajele criptate este vecinul tău, tot este mai puţin sigur decât dacă nu ţi le citea nimeni.

Sistemul telefonic este în general mai de încredere decât cablul. Dispune, de exemplu, de un sistem suplimentar de alimentare cu energie în caz de urgenţă şi funcţionează normal chiar şi în timpul unei pene de curent. La cablu, dacă energia pentru orice amplificator de-a lungul lanţului cade, toţi utilizatorii de după el sunt deconectaţi instantaneu.

În sfârşit, cei mai mulţi furnizori ADSL oferă posibilitatea alegerii unui ISP. Uneori li se cere acest lucru chiar prin lege. Nu este întotdeauna cazul cu operatorii prin cablu.

Concluzia este că ADSL şi cablul sunt mai mult asemănătoare decât diferite. Oferă servicii comparabile şi, cum competiţia dintre ei se încinge, probabil că vor oferi şi preţuri comparabile.

2.8 REZUMAT

Nivelul fizic stă la baza tuturor reţelelor. Natura impune două limite fundamentale asupra unui canal, iar acestea determină lărgimea de bandă. Este vorba despre limita Nyquist, care se aplică asupra canalelor fără zgomot, şi limita Shannon, pentru canale cu zgomot.

Mediile de transmisie pot fi ghidate sau neghidate. Principalele medii ghidate sunt cablul torsadat, cablul coaxial şi fibra optică. Mediile neghidate includ undele radio, microundele, undele în în- fraroşu şi razele laser care se propagă prin aer. Un sistem de transmisie readus în actualitate este comunicaţia prin satelit, în special sistemele LEO.

Elementul cheie pentru majoritatea reţelelor pe arii geografic largi este sistemul

telefonic. Componentele sale principale sunt buclele locale, trunchiurile şi comutatoarele. Buclele locale sunt circuite analogice de cablu torsadat, care necesită folosirea modemurilor pentru transmisia datelor digitale. ADSL oferă viteze de până la 50 Mbps prin divizarea buclei locale în mai multe canale virtuale şi modularea fiecăruia separat. Buclele locale fără fir reprezintă o noua direcţie de dezvoltare de urmărit, în special LMDS.

Trunchiurile sunt digitale şi pot fi multiplexate în mai multe moduri, printre care: FDM, TDM şi WDM. Atât comutarea de circuite cât şi comutarea de pachete reprezintă tehnologii importante.

Pentru aplicaţiile mobile, sistemul de telefonie fixă (cu fir) nu este adecvat. Radioul celular este folosit tot mai mult pentru transmisia de voce, iar în curând va fi folosit frecvent şi pentru schimbul de date. Prima generaţie de mobile a fost analogică, dominată de AMPS. A doua generaţie a fost digitală, cu D-AMPS, GSM şi CDMA ca opţiuni majore. A treia generaţie va fi digitală şi bazată pe CDMA de bandă largă.

Un sistem alternativ pentru accesul la reţea este sistemul de televiziune prin cablu, care a evoluat gradat de la o antenă colectivă la un sistem hibrid fibra optică-coax. Potenţial, acesta oferă lărgime de bandă foarte mare, dar lărgimea de bandă reală disponibilă depinde semnificativ de numărul de utilizatori activi la un moment dat şi de ceea ce fac ei.

2.9 PROBLEME

1. Calculaţi coeficienţii Fourier pentru funcţia $f(t) = t$, $(0 < t < 1)$.

2. Un canal de 4 KHz, fără zgomot, este eşantionat la fiecare 1 msec. Care este rata maximă de transfer a datelor?

3. Canalele de televiziune au o lăţime de 6 MHz. Câţi biţi/sec pot fi transmişi dacă se folosesc semnale digitale pe patru niveluri? Consideraţi cazul unui canal fără zgomot.

4. Dacă un semnal binar este transmis pe un canal de 3 KHz al cărui raport semnal-zgomot este de 20dB, care este rata maximă de transfer a datelor ce se poate realiza?

5. Ce raport semnal-zgomot este necesar pentru a pune o purtătoare T1 pe o linie de 50 KHz?

6. Care este diferenţa dintre o stea pasivă şi un repetor activ într-o reţea pe fibră optică?

7. Care este lărgimea de bandă existentă în 0,1 microni de spectru la o lungime de undă de 1 micron?

8. Se doreşte să se transmită prin fibră optică o secvenţă de imagini de pe ecranul calculatorului. Ecranul are 480 x 640 pixeli, fiecare pixel având 24 biţi. Există 60 imagini ecran pe secundă. Ce lăţime de bandă este necesară şi care este lungimea de undă necesară pentru această bandă la 1,30 microni?

9. Teorema lui Nyquist este adevărată şi pentru fibra optică sau numai pentru cablul de cupru?

10. În fig. 2-6 banda din partea stângă este mai îngustă decât celelalte. De ce?

Deseori, antenele radio funcţionează cel mai bine atunci când diametrul antenei este egal cu lungimea de undă a undei radio. Antenele rezonabile au între 1 cm şi 5 m în diametru. Ce domeniu de frecvenţă acoperă acestea?

Atenuarea multi-căi este maximizată atunci când două raze sosesc cu un defazaj de 180 grade. Cât de mare trebuie să fie diferenţa de drum pentru a maximiza atenuarea în cazul unei legături prin microunde de 1 GHz având 50 km lungime?

O rază laser de 1 mm lăţime este urmărită de un detector de 1 mm lăţime aflat la 100 m distanţă, pe acoperişul unei clădiri. Care este limita maximă a deviaţiei unghiulare (în grade) a laserului pentru care raza poate fi captată de detector ?

Cei 66 de sateliţi de joasă altitudine din proiectul Iridium sunt împărţiţi în 6 coliere în jurul Pământului. La altitudinea la care sunt folosiţi, perioada de rotaţie este de 90 minute. Care este intervalul mediu pentru timpii morţi (de inactivitate) în cazul unui emiţător staţionar?

Se considera un satelit la altitudinea sateliţilor geostaţionari, dar al cărui plan orbital este înclinat faţă de planul ecuatorului cu un unghi 4>- Un utilizator staţionar se află pe suprafaţa pământului la latitudinea nordică 4>. Este adevărat că acestui utilizator i se pare că satelitul este nemişcat pe cer? Dacă nu, descrieţi mişcarea pe care o percepe.

Câte coduri de oficiu final erau înainte de 1984, atunci când fiecare oficiu final era identificat după codul său de zonă, format din 3 cifre, combinat cu primele trei cifre ale numărului local ? Codurile de zonă începeau cu o cifra din intervalul 2-9, avea a doua cifră un 0 sau un 1 şi se puteau termina cu orice cifră. Primele două cifre din numărul local erau întotdeauna din intervalul 2-9. A treia cifră putea fi oricare.

Folosind *numai* datele din text, care este numărul maxim de telefoane pe care sistemul existent în SUA le poate suporta fără a se schimba numerotaţia şi fără a se adăuga echipament adiţional? Ar putea fi atins acest număr în realitate? Pentru simplificarea ipotezei, un calculator sau un fax este numărat tot ca un telefon. Presupuneţi că exista un singur dispozitiv pentru o linie de abonat.

Un sistem telefonic simplu este alcătuit din două oficii finale şi un singur oficiu de taxare, la care fiecare oficiu final este conectat printr-un trunchi duplex de 1 MHz. Un telefon obişnuit este folosit pentru a face 4 apeluri într-o zi lucrătoare de 8 ore. Durata medie a unui apel este de 6 minute. 10% dintre apeluri sunt de distanţă lungă (adică traversează oficiul de taxare). Care este numărul maxim de telefoane pe care îl poate suporta un oficiu final? (presupuneţi 4 KHz pe circuit)

O companie regională de telefoane are 10 milioane de abonaţi. Fiecare dintre telefoanele acestora este conectat la un oficiu central printr-un cablu torsadat

de cupru. Lungimea medie a acestor cabluri este de 10 km. Cât de mult reprezintă cuprul din valoarea buclelor locale? Presupuneţi că secţiunea transversală a fiecărui fir este de 1 mm diametru, greutatea specifică a cuprului este 9,0 şi cuprul se vinde cu 3 dolari pe kg.

O conductă de petrol este un sistem simplex, half duplex, full duplex sau nici una dintre variantele menţionate?

21. Costul unui microprocesor rapid a scăzut într-atât încât este posibil să se includă câte unul în fiecare modem. Cum afectează aceasta gestiunea erorilor liniei telefonice?

22. O diagramă-constelaţie a unui modem, similară celei din fig. 2-25, are puncte la următoarele coordonate: (1,1), (1,-1), (-l,1) şi (-l,-1)- Câţi biţi pe secundă poate atinge un modem cu aceşti parametri, la 1200 baud ?

23. O diagramă-constelaţie a unui modem, similară celei din fig. 2-25, are puncte în (0,1) şi (0,2). Modemul foloseşte modulaţie în fază sau modulaţie în amplitudine?

24. Într-o diagramă-constelaţie, toate punctele sunt situate pe un cerc centrat în origine. Ce fel de modulaţie se foloseşte ?

25. Câte frecvenţe foloseşte un modem full-duplex QAM-64?

26. Un sistem ADSL care foloseşte DMT alocă 3/4 dintre canalele de date disponibile pentru legătura de flux descendent. Se foloseşte o modulaţie QAM-64 pe fiecare canal. Care este capacitatea legăturii de flux descendent?

27. În exemplul de LMDS cu patru sectoare din fig. 2-30, fiecare sector are propriul său canal de 36 Mbps. Potrivit teoriei cozilor, dacă un canal este 50% plin, timpul de aşteptare va fi egal cu timpul de transfer descendent. În aceste condiţii, cât durează să se transfere o pagină Web de 5 KB ? Cât timp durează să se transfere aceeaşi pagină pe o linie ADSL de 1 Mbps ? Dar cu un modem de 56 Kbps ?

28. Zece semnale, fiecare necesitând 4000 Hz, sunt multiplexate în acelaşi canal utilizând FDM. Care este lăţimea de bandă minimă necesară pentru canalul multiplexat? Presupunem că benzile suplimentare (de gardă) au lăţimea de 400 Hz.

29. De ce a fost stabilit timpul de eşantionare PCM la 125 psec?

30. Care este procentul de supraîncărcare pe o purtătoare T1? Mai exact, ce procent din cei 1.544 Mbps nu este pus la dispoziţia utilizatorului final?

31. Comparaţi rata maximă de transfer al datelor pe un canal de 4 KHz, fără zgomot, care foloseşte:

a) codificare analogică (de ex. QPSK) cu 2 biţi pe eşantion;
b) sistemul PCM T1.

32. Dacă un sistem cu purtătoarea T1 se desincronizează şi pierde tactul, el încearcă să se resincro- nizeze folosind primul bit din fiecare cadru. Câte cadre vor trebui inspectate, în medie, pentru a se resincroniza cu o probabilitate de eşec de 0,001?

33. Care este diferenţa, dacă există vreuna, între blocul de demodulare a unui modem şi blocul de codificare a unui codec? (În definitiv, ambele convertesc

semnale analogice în semnale digitale.)

34. Un semnal este transmis digital pe un canal de 4 KHz, fără zgomot, cu un eşantion la fiecare 125 ps. Câţi biţi sunt de fapt transmişi pe secundă pentru fiecare dintre aceste metode de codificare:

a) Standardul CCITT de 2,048 Mbps;
b) DPCM cu o valoare relativă a semnalului pe 4 biţi;
c) Modulaţia delta.

Un semnal pur sinusoidal de amplitudine A este codificat folosind modulaţia delta, cu x eşan- tioane/secundă. Un semnal de ieşire de +1 corespunde unei schimbări a semnalului cu $+A/8$ iar un semnal de ieşire de -l corespunde unei schimbări a semnalului cu $-A/8$. Care este cea mai mare frecvenţă care poate fi urmărită fără erori cumulative?

Ceasurile SONET au o rată de deviaţie de aproximativ $1/10^9$. Cât timp este necesar pentru ca deviaţia să egaleze lăţimea unui bit? Care sunt implicaţiile acestui calcul?

În fig. 2-37 rata de transfer a datelor utilizatorului pentru OC-3 a fost stabilită la 148,608 Mbps. Arătaţi cum poate fi obţinut acest număr din parametrii SONET OC-3.

Pentru a fi integra rate de transmisie a datelor mai mici decât STS-l, SONET are un sistem de fluxuri parţiale virtuale (VT, eng. Virtual Tributaries). O VT este o încărcătură utilă parţială care poate fi inserată într-un cadru STS-l şi combinată cu alte încărcături parţiale pentru a completa un cadru de date. VT1.5 foloseşte 3 coloane, VT2 foloseşte 4 coloane, VT3 foloseşte 6 coloane, iar VT6 foloseşte 12 coloane dintr-un cadru STS-1. Care dintre VT poate integra:

a) un serviciu DS-l (1.544 Mbps) ?
b) serviciul european CEPT-l (2.048 Mbps) ?
c) un serviciu DS-2 (6.312 Mbps) ?

Care este diferenţa esenţială dintre comutarea de mesaje şi comutarea de pachete ?

Care este lăţimea de bandă disponibilă utilizatorului într-o conexiune OC-12c?

Se dau trei reţele cu comutare de pachete, conţinând fiecare câte n noduri. Prima reţea are o topologie stea cu un comutator central, cea de-a doua este un inel (bidirecţional), iar cea de-a treia este interconectată complet, având câte o legătură de la fiecare nod către toate celelalte noduri. Care sunt lungimile căilor, măsurate în salturi (între noduri), pentru cazul cel mai bun, pentru cazul mediu şi pentru cazul cel mai defavorabil?

Comparaţi întârzierea în transmisia unui mesaj de x biţi pe o cale de k salturi (între noduri) dintr-o reţea cu circuite comutate şi într-o reţea cu comutare de pachete (puţin aglomerată). Timpul de stabilire a circuitului este S sec, întârzierea de propagare este d sec/salt, dimensiunea pachetului este de p biţi şi rata de transfer a datelor este b biţi/sec. În ce condiţii reţeaua cu comutare de pachete are o întârziere mai mică?

Presupunem că x biţi de date ale utile trebuie transmişi pe o cale de k salturi (între noduri), într- o reţea cu comutare de pachete, ca o serie de pachete; fiecare pachet conţine p biţi de date şi h biţi pentru antet, cu $x >> (p+h)$. Rata de transfer a liniei este de b bps şi întârzierea de propagare este neglijabilă. Ce valoare a lui p minimizează întârzierea totală?

Într-un sistem tipic de telefonie mobilă cu celule hexagonale, este interzis să se refolosească banda de frecvenţe a unei celule într-o celulă adiacentă. Dacă sunt disponibile în total de 840 frecvenţe, câte frecvenţe se pot folosi într-o anumită celulă?

Aranjamentul real a celulelor este rareori atât de regulat precum cel din fig. 2-41. Până şi formele celulelor individuale sunt de obicei neregulate. Daţi un motiv plauzibil pentru acest fapt.

46. Faceţi o estimare sumară a numărului de microcelule PCS de 100 m diametru, care ar fi necesare pentru a acoperi San Francisco (120 km^2).

47. Uneori, atunci când un utilizator traversează graniţa dintre două celule, apelul curent se termină brusc, deşi toate emiţătoarele şi receptoarele funcţionează perfect. De ce?

48. D-AMPS oferă o calitate a vocii sensibil mai slabă decât GSM. Este adevărat că acest fapt se datorează cerinţei ca D-AMPS să păstreze compatibilitatea cu AMPS, în timp ce GSM nu a avut astfel de constrângeri? Dacă nu, care este cauza ?

49. Calculaţi numărul maxim de utilizatori pe care D-AMPS îi poate suporta simultan într-o singură celulă. Faceţi acelaşi calcul pentru GSM. Explicaţi diferenţa.

50. Să presupunem că A, B şi C transmit simultan biţi 0 folosind sistemul CDMA cu secvenţele de felii din fig. 2-45(b). Care este secvenţa de felii rezultată?

51. În discuţia despre ortogonalitatea secvenţelor de felii CDMA, s-a afirmat că dacă $\mathbf{S} \cdot \mathbf{T} = 0$ atunci şi $\mathbf{S} \cdot \mathbf{T} = 0$. Demonstraţi acest fapt.

52. Consideraţi un alt mod de a privi proprietatea de ortogonalitate a secvenţelor de felii CDMA: fiecare bit dintr-o pereche de secvenţe se poate potrivi sau nu. Exprimaţi proprietatea de orto- gonalitate în termeni de potriviri şi nepotriviri.

53. Un receptor CDMA primeşte următoarele felii : (-l +1 -3 +1 -l -3 +1 +1). Folosind secvenţele de felii definite în fig. 2-45 (b), care dintre staţii au transmis şi ce biţi a transmis fiecare?

54. La nivelul inferior, sistemul telefonic este construit în formă de stea, cu toate buclele locale dintr-un cartier convergente către un oficiu final. Din contră, televiziunea prin cablu este alcătuită dintr-un singur cablu lung, cu un traseu şerpuit pe deasupra tuturor caselor din acelaşi cartier. Presupunem că în viitor cablul TV va fi din fibră optică de 10 Gbps în loc de cupru. Ar putea acesta fi folosit pentru a simula modelul telefonic în care fiecare să aibă propria sa linie către oficiul final? Dacă da, câte case cu un telefon pot fi conectate la o singură fibră optică?

55. Un sistem de TV prin cablu are 100 de canale comerciale, fiecare din acestea alternând programele cu publicitatea. Această organizare seamănă cu TDM sau cu FDM ?

56. O companie de cablu decide să ofere acces Internet prin cablu într-un cartier cu 5000 de case. Compania foloseşte un cablu coaxial şi o alocare de spectru care oferă o lăţime de bandă de 100 Mbps pentru flux descendent pentru fiecare cablu. Pentru a atrage clienţii, compania decide să garanteze cel puţin 2 Mbps lăţime de bandă pentru flux descendent pentru fiecare casă, în orice moment. Descrieţi modul în care trebuie să acţioneze compania de cablu pentru a oferi

această garanţie.

57. Utilizând alocarea spectrală ilustrată în fig. 2-48 şi informaţiile date în text, câţi Mbps alocă un sistem de cablu pentru fluxul ascendent şi câţi pentru cel descendent ?

58. Cât de repede poate un utilizator de cablu să primească date dacă reţeaua este în rest inactivă ?

59. Multiplexarea fluxurilor multiple de date STS-l, numite fluxuri parţiale (eng: Tributaries), joacă un rol important în SONET. Un multiplexor 3:1 multiplexează trei fluxuri parţiale STS-l primite la intrare într-un flux de ieşire STS-3. Multiplexarea este făcută octet cu octet, adică primii trei octeţi de ieşire sunt primii octeţi ai fluxurilor parţiale 1, 2 respectiv 3. Următorii octeţi de ieşire alcătuiesc al doilea grup de fluxuri 1, 2 şi 3, şi aşa mai departe. Scrieţi un program care să simuleze acest multiplexor 3:1. Programul va conţine 5 procese. Procesul principal creează patru alte procese, câte unul pentru fiecare dintre cele 3 fluxuri parţiale STS-l şi unul pentru multiplexor. Fiecare proces de tip flux parţial citeşte un cadru STS-l din fişierul de intrare ca pe o succesiune de 810 biţi şi îşi trimite cadrul octet cu octet procesului multiplexor. Procesul multiplexor recepţionează octeţii şi furnizează la ieşire un cadru STS-3 (tot octet cu octet) prin afişarea pe ecran. Pentru comunicaţia între procese folosiţi conducte (eng: pipe).

NIVELUL LEGĂTURĂ DE DATE

În acest capitol vom studia arhitectura nivelului 2, nivelul legătură de date. Acest studiu se ocupă de algoritmii de obţinere a unei comunicaţii eficiente şi sigure, între două maşini adiacente la nivelul legăturii de date. Prin adiacenţă înţelegem că cele două maşini sunt conectate fizic printr-un canal de comunicaţie care se manifestă conceptual ca un fir (de exemplu, un cablu coaxial, o linie telefonică sau un canal de comunicaţie fără fir, de tip punct la punct). Calitatea esenţială a unui canal care îl face asemănător unui fir este aceea că biţii sunt livraţi în exact aceeaşi ordine în care sunt transmişi.

La început aţi putea crede că această problemă este atât de simplă, încât nu există programe de analizat – maşina A pune biţii pe fir şi maşina B îi preia. Din păcate, circuitele de comunicaţie produc uneori erori. În plus, ele au numai o rată finită a datelor şi există o întârziere a propagării, nenulă, între momentul în care un bit este emis şi momentul în care acesta este recepţionat. Aceste limitări au implicaţii importante pentru eficienţa transferului de date. Protocoalele utilizate pentru comunicaţie trebuie să ia în considerare toţi aceşti factori. Aceste protocoale reprezintă subiectul capitolului de faţă.

După o introducere în principalele aspecte ale proiectării nivelului legătură de date, vom începe studiul protocoalelor examinând natura erorilor, cauzele producerii lor şi cum pot fi ele detectate şi corectate. Apoi vom studia o serie de protocoale din ce în ce mai complexe, fiecare dintre ele rezolvând cât mai multe dintre problemele prezente la acest nivel. Vom încheia cu un studiu al modelării şi corectitudinii protocoalelor şi vom da câteva exemple de protocoale ale legăturii de date.

3.1 ASPECTE ALE PROIECTĂRII NIVELULUI LEGĂTURĂ DE DATE

Nivelul legătură de date are un număr de funcţii specifice pe care trebuie să le îndeplinească. Aceste funcţii includ:

1. Furnizarea unei interfeţe bine-definite către nivelul reţea
2. Tratarea erorilor de transmisie
3. Reglarea fluxului cadrelor în aşa fel, încât receptorii lenţi să nu fie inundaţi de către emiţători rapizi.

Pentru a îndeplini aceste scopuri, nivelul legătură de date primeşte pachete de la nivelul reţea, pe care le încapsulează în **cadre** în vederea transmiterii. Fiecare cadru conţine un antet, un câmp de informaţie utilă pentru pachet şi încheiere, după cum se vede în fig. 3-1. Gestionarea cadrelor reprezintă esenţa a ceea ce face nivelul legătură de date. În secţiunile următoare vom examina în detaliu toate aspectele menţionate anterior.

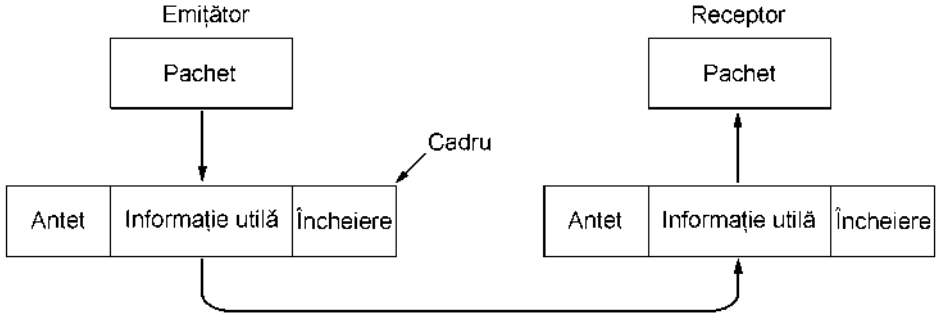


Fig. 3-1. Relaţia dintre pachete şi cadre.

Cu toate că acest capitol se referă numai la nivelul legătură de date şi la protocoalele legăturii de date, multe dintre principiile pe care le vom studia aici, cum ar fi controlul erorilor şi controlul fluxului, se regăsesc şi în protocoalele de transport, şi în alte protocoale. În realitate, în multe reţele, aceste funcţii se găsesc doar la nivelurile superioare, nu şi la nivel legătură de date. Oricum, indiferent de nivelul la care se găsesc, principiile sunt aproximativ aceleaşi, deci nu contează prea mult unde le studiem. La nivelul legăturii de date ele apar de obicei în forma cea mai simplă şi cea mai pură, făcând din acest nivel un loc foarte potrivit studierii detaliate a acestor principii.

3.1.1 Servicii oferite nivelului reţea

Funcţia nivelului legătură de date este să ofere servicii nivelului reţea. Principalul serviciu este transferul datelor de la nivelul reţea al maşinii sursă la nivelul reţea al maşinii destinaţie. La nivelul reţea al maşinii sursă există o entitate,

să-i spunem proces, care trimite biţi către nivelul legătură de date, pentru a fi transmişi la destinaţie. Funcţia nivelului legătură de date este să transmită biţii spre maşina destinaţie, pentru ca acolo să fie livraţi nivelului reţea, aşa cum se arată în fig. 3-2(a). Transmisia efectivă urmează calea din fig. 3-2(b), dar este mai uşor de înţeles în termenii a două procese

ale nivelului legătură de date care comunică utilizând un protocol al legăturii de date. Din acest motiv, pe parcursul acestui capitol, vom folosi în mod implicit modelul din fig. 3-1(a).

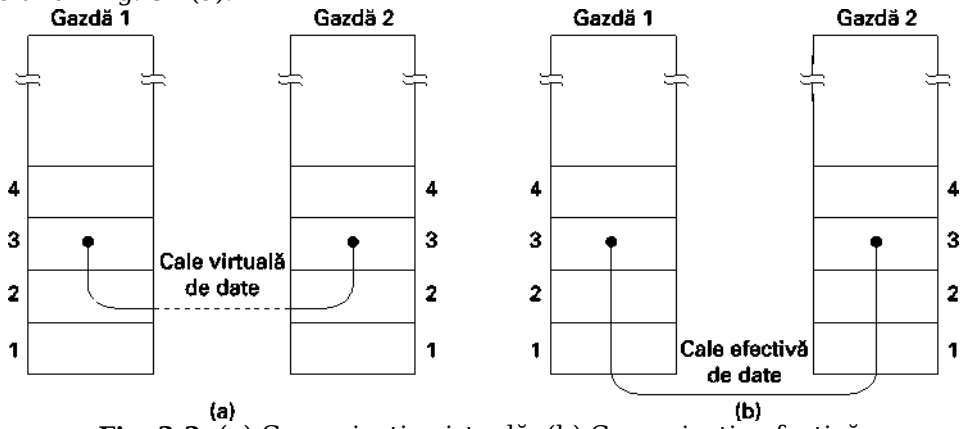


Fig. 3-2. (a) Comunicaţie virtuală. (b) Comunicaţie efectivă.

Nivelul legătură de date poate fi proiectat să ofere diferite servicii. Serviciile efective oferite pot varia de la sistem la sistem. Trei posibilităţi de bază, oferite în mod curent, sunt:

1. Serviciu neconfirmat fără conexiune.
2. Serviciu confirmat fără conexiune.
3. Serviciu confirmat orientat-conexiune.

Să le analizăm pe rând pe fiecare dintre acestea.

Serviciul neconfirmat fără conexiune constă din aceea că maşina sursă trimite cadre independente către maşina destinaţie, fără ca maşina destinaţie să trebuiască să confirme primirea lor. În acest caz, nu sunt necesare stabilirea şi desfiinţarea unei conexiuni logice. Dacă un cadru este pierdut datorită zgomotului de pe linie, la nivelul legătură de date nu se face nici o încercare pentru recuperarea lui. Această clasă de servicii este adecvată atunci când rata de erori este foarte scăzută, încât recuperarea este lăsată în sarcina nivelurilor superioare. De asemenea, este adecvată pentru traficul de timp real, cum ar fi cel de voce, unde a primi date cu întârziere este mai rău decât a primi date eronate. Majoritatea LAN-urilor utilizează la nivelul legăturii de date servicii neconfirmate fără conexiune.

Următorul pas în ceea ce priveşte siguranţa este serviciul confirmat fără conexiune. Atunci când este oferit acest serviciu, încă nu se utilizează conexiuni, dar fiecare cadru trimis este confirmat individual. În acest mod, emiţătorul ştie dacă un cadru a ajuns sau nu cu bine. Dacă nu a ajuns într-un interval de timp

specificat, poate fi trimis din nou. Acest serviciu este folositor pentru canale nesigure, cum ar fi sistemele fără fir.

Poate că merită să subliniem că asigurarea confirmării la nivelul legăturii de date este doar o optimizare, niciodată o cerinţă. Nivelul reţea poate întotdeauna să trimită un mesaj şi să aştepte să fie confirmat. Dacă confirmarea nu apare în timp util, atunci emiţătorul poate retrimite întregul mesaj.

Problema cu această strategie este aceea că, de obicei, cadrele au o lungime maximă impusă de hardware, iar pachetele nivelului reţea nu au aceasta limitare. Dacă pachetul mediu este spart în, să zicem, 10 cadre şi 20% din totalul cadrelor sunt pierdute, transmiterea acestuia poate lua foarte mult timp. În cazul în care cadrele individuale sunt confirmate şi retransmise, pachetele întregi vor fi transmise mult mai rapid. Pe canale sigure, precum fibra optică, costul suplimentar implicat de un astfel de protocol al legăturii de date poate fi nejustificat, dar pe canale fără fir, costul este pe deplin justificat datorită nesiguranţei acestora.

Revenind la serviciile noastre, cel mai sofisticat serviciu pe care nivelul legătură de date îl pune la dispoziţia nivelului reţea este serviciul orientat-conexiune. În cazul acestui serviciu, maşinile sursă şi destinaţie stabilesc o conexiune înainte de a transfera date. Fiecare cadru trimis pe conexiune este numerotat şi nivelul legătură de date garantează că fiecare cadru trimis este într-adevăr recepţionat. Mai mult, garantează că fiecare cadru este recepţionat exact o dată şi toate cadrele sunt recepţionate în ordinea corectă. În schimb, în cazul serviciului fără conexiune, este posibil ca, datorită unei confirmări pierdute, un cadru să fie transmis de mai multe ori şi, prin urmare, recepţionat de mai multe ori. Spre deosebire de acesta, serviciul orientat conexiune furnizează proceselor de la nivelul reţea echivalentul unui flux de biţi sigur.

Atunci când este utilizat serviciul orientat conexiune, transferurile au trei faze distincte. În prima fază este stabilită conexiunea, ambele părţi iniţializând variabile şi contoare, utilizate pentru a ţine evidenţa cadrelor care au fost recepţionate şi a celor care nu au fost. În a doua fază, sunt transmise unul sau mai multe cadre. În a treia şi ultima fază, conexiunea este desfiinţată, eliberând variabilele, tampoanele şi alte resurse utilizate la menţinerea conexiunii.

Să considerăm un exemplu tipic: o subreţea WAN formată din rutere conectate prin linii telefonice punct-la-punct, închiriate. Când un cadru ajunge la un ruter, hardware-ul verifică absenţa erorilor (folosind tehnici pe care le vom studia mai târziu în acest capitol), şi trimite cadrul programelor nivelului legătură de date (care se pot afla într-un cip de pe adaptorul de reţea). Programele nivelului legătură de date verifică dacă acesta este cadrul aşteptat şi, dacă este aşa, trimit pachetul din câmpul de informaţie utilă către programele de dirijare. Programele de dirijare aleg linia de ieşire corespunzătoare şi trimit pachetul înapoi, către programele nivelului legătură de date, care apoi îl transmit. Fluxul dintre două rutere este reprezentat în fig. 3-3.

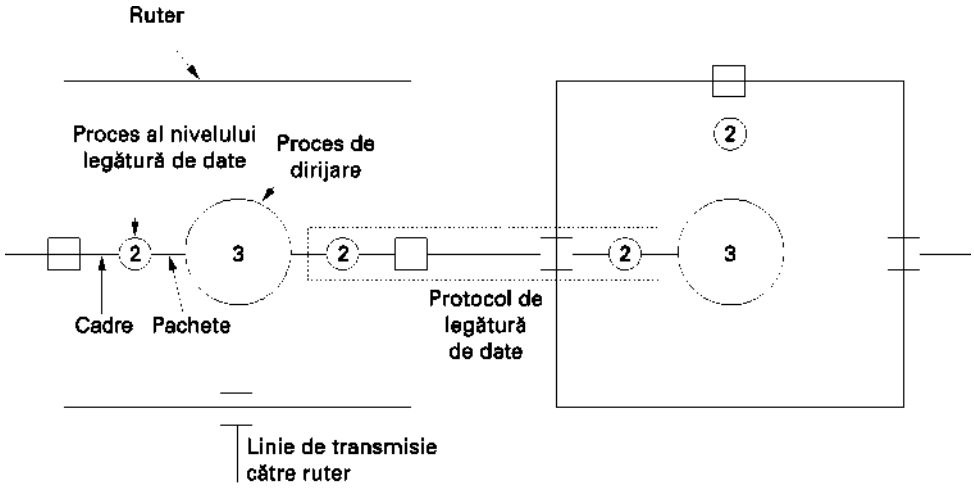


Fig. 3-3. Plasarea protocolului legătură de date.

Programul de dirijare doreşte de obicei ca operaţia să fie corect executată, ceea ce presupune conexiuni secvenţiale sigure pe fiecare linie punct-la-punct. El nu vrea să fie deranjat prea des de pachete care s-au pierdut pe drum. Este sarcina protocolului legăturii de date, prezentat în dreptun

ghiul punctat, să facă liniile de comunicaţie nesigure să pară perfecte sau, cel puţin, suficient de bune. Această proprietate este foarte importantă pentru legăturile fără fir, care sunt, prin natura lor, foarte nesigure. Ca o remarcă, cu toate că am prezentat copii ale programelor nivelului legătură de date în fiecare ruter, de fapt există o singură copie, care tratează toate liniile, utilizând tabele şi structuri de date diferite pentru fiecare dintre ele.

3.1.2 Încadrarea

În vederea furnizării unui serviciu nivelului reţea, nivelul legătură de date trebuie să utilizeze serviciul furnizat de către nivelul fizic. Sarcina nivelului fizic este să primească un flux de biţi şi să încerce să-l trimită la destinaţie. Nu se garantează că acest flux de biţi nu conţine erori. Numărul de biţi recepţionaţi poate fi mai mic, egal cu, sau mai mare decât numărul de biţi transmişi şi pot avea valori diferite. Este la latitudinea nivelului legătură de date să detecteze şi, dacă este necesar, să corecteze erorile.

Abordarea uzuală pentru nivelul legătură de date este să spargă şirul de biţi în cadre discrete şi să calculeze suma de control pentru fiecare cadru. (Algoritmii pentru suma de control vor fi discutaţi mai târziu în acest capitol.) Atunci când un cadru ajunge la destinaţie, suma de control este recalculată. Dacă noua sumă de control este diferită de cea conţinută în cadru, nivelul legătură de date ştie că a apărut o eroare şi face operaţiile necesare pentru a o rezolva (de exemplu, elimină cadrul eronat şi probabil trimite înapoi un raport de eroare).

Spargerea şirului de biţi în cadre este mai dificilă decât pare la prima vedere. O cale pentru a realiza această încadrare este inserarea de intervale de timp între cadre, aşa cum inserăm spaţii între cuvinte într-un text normal. Totuşi, reţelele dau rareori garanţii referitoare la timp, aşa că este posibil ca aceste intervale să fie comprimate sau ca în timpul transmisiei să fie inserate alte intervale.

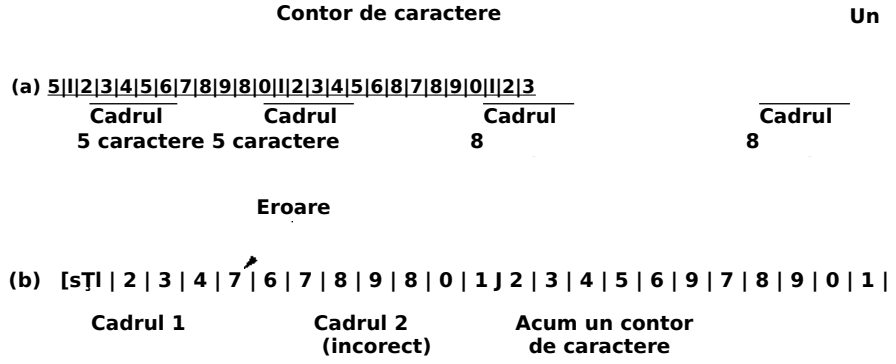

Contor de caractere
Un
(a) 5|l|2|3|4|5|6|7|8|9|8|0|l|2|3|4|5|6|8|7|8|9|0|l|2|3
Cadrul
5 caractere
Cadrul
5 caractere
Cadrul
8
Cadrul
8
Eroare
(b) [sŢl | 2 | 3 | 4 | 7 | 6 | 7 | 8 | 9 | 8 | 0 | 1 J 2 | 3 | 4 | 5 | 6 | 9 | 7 | 8 | 9 | 0 | 1 | 2 | 3 |
Cadrul 1
Cadrul 2
(incorect)
Acum un contor
de caractere

Fig. 3-4. Un şir de caractere. (a) Fără erori. (b) Cu o eroare.

Deoarece este prea periculos să ne bizuim pe timp pentru a marca începutul şi sfârşitul fiecărui cadru, au fost elaborate alte metode. În această secţiune vom

analiza patru metode:

1. Numărarea caracterelor.
2. Indicatori cu inserare de octeţi.
3. Indicatori de început şi de sfârşit, cu inserare de biţi.
4. Violarea codificărilor la nivel fizic.

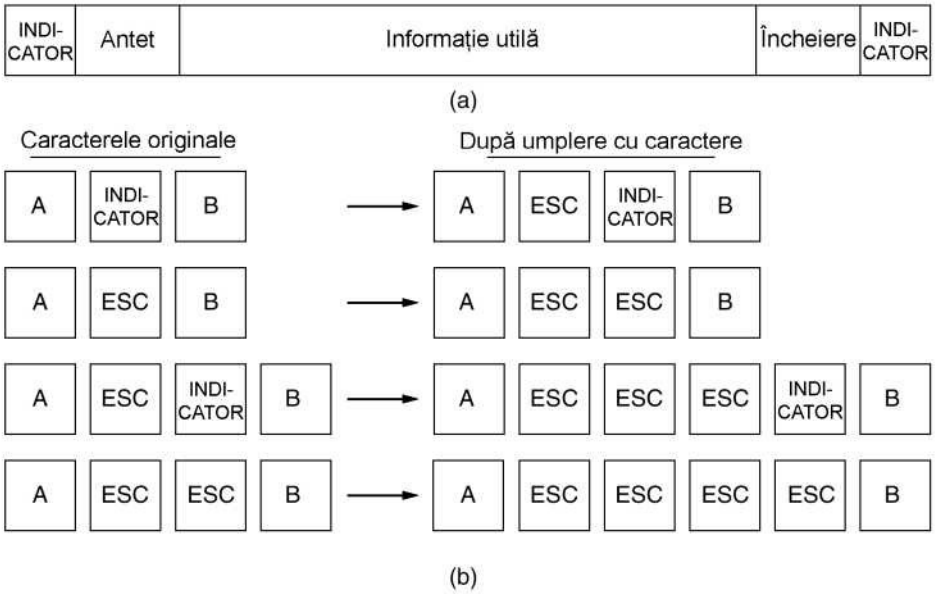


Fig. 3-5. (a) Cadru delimitat de octeţi indicatori. (b) Patru exemple de secvenţe de octeţi înainte şi după inserare

Prima metodă de încadrare utilizează un câmp din antet pentru a specifica numărul de caractere din cadru. Atunci când nivelul legătură de date de la destinaţie primeşte contorul de caractere, ştie câte caractere urmează şi unde este sfârşitul cadrului. Această tehnică este prezentată în fig. 3-4 (a) pentru patru cadre de dimensiune de 5, 5, 8 şi 8 caractere.

Problema cu acest algoritm este că valoarea contorului poate fi alterată de erori de transmisie. De exemplu, dacă contorul de caractere din al doilea cadru din fig. 3-4(b) din 5 devine 7, destinaţia va pierde sincronizarea şi va fi incapabilă să localizeze începutul cadrului următor. Chiar dacă suma de control este incorectă şi destinaţia ştie că a primit cadru eronat, nu există nici o posibilitate de a determina unde începe următorul cadru. Nu ajută nici trimiterea unui cadru înapoi la sursă, cerând o retransmisie, deoarece destinaţia nu ştie peste câte caractere să sară pentru a începe retransmisia. Din acest motiv, metoda contorizării caracterelor este rar utilizată.

A doua metodă de încadrare înlătură problema resincronizării după o eroare, prin aceea că fiecare cadru începe şi se termină cu o secvenţa specială de octeţi. Iniţial, octeţii ce indicau începutul, respectiv sfârşitul erau diferiţi, dar în ultimii ani s-a trecut la utilizarea unui singur octet, numit octet indicator, atât ca indicator de

început, cât şi de sfârşit, aşa cum se prezintă în fig. 3-5(a). În acest fel, dacă receptorul pierde sincronizarea, acesta poate căuta octetul indicator pentru a găsi sfârşitul cadrului. Doi octeţi indicatori consecutivi indică sfârşitul unui cadru şi începutul celui care urmează.

O problemă serioasă cu această metodă apare atunci când se transmit date binare, cum ar fi un obiect sau numere în virgulă mobilă. Se poate întâmpla ca în date să apară octetul folosit ca indicator. Această situaţie interferează cu procesul de încadrare. O cale de rezolvare a acestei probleme este ca nivelul legătură de date al emiţătorului să insereze un octet special (ESC) înaintea fiecărei apariţii „accidentale" a indicatorului în date. Nivelul legătură de date al receptorului va elimina acest octet special înainte de a pasa datele nivelului reţea. Această tehnică poartă numele de **inserare de octeţi** (eng.: **byte stufflng**) sau **inserare de caractere** (eng.: **character stufflng**) . Deci, un octet indicator utilizat pentru încadrare poate fi diferenţiat de unul prezent în date prin faptul că este sau nu precedat de un octet special.

Bineînţeles, următoarea întrebare este: Ce se întâmplă dacă un octet special apare în mijlocul datelor? Răspunsul este că pe lângă el se inserează încă un octet special.

Deci, un singur octet special face parte dintr-o secvenţă specială, iar un octet special dublu indica faptul ca un singur octet de acest tip a apărut în cadrul datelor. Câteva exemple sunt prezentate în fig. 3-5(b). În toate cazurile, secvenţa de octeţi obţinută după eliminarea octeţilor inseraţi este exact secvenţa originală.

Schema de inserare de octeţi prezentată în fig. 3-5 reprezintă o uşoară simplificare a schemei utilizate în cadrul protocolului PPP, pe care majoritatea calculatoarelor casnice îl folosesc pentru a comunica cu furnizorul de servicii Internet. Vom discuta despre PPP mai târziu în acest capitol.

Un dezavantaj major al utilizării acestei metode de încadrare este acela că este limitată la utilizarea caracterelor de 8 biţi. Nu toate codurile utilizează caractere de 8 biţi. De exemplu, UNICODE foloseşte caractere pe 16 biţi. Datorită dezvoltării reţelelor, dezavantajele inserării de coduri de caractere în mecanismul de încadrare au devenit din ce în ce mai evidente, aşa că a trebuit dezvoltată o nouă tehnică, care să permită caractere de dimensiune variabilă.

Noua tehnică permite cadrelor de date să conţină un număr arbitrar de biţi şi permite coduri de caractere cu un număr arbitrar de biţi per caracter. Funcţionează astfel: fiecare cadru începe şi se termină cu un şablon special pe biţi, 01111110, numit octet **indicator (flag)**. De fiecare dată când nivelul legătură de date al emiţătorului identifică cinci de unu consecutivi în date, inserează automat un bit 0 în şirul de biţi de rezultaţi. Această **inserare de biţi (bit stuffing)** este similară inserării de caractere, în care un octet escape este inserat în şirul de caractere de ieşire, înainte de fiecare octet indicator din date.

Atunci când receptorul primeşte o succesiune de cinci biţi 1, urmaţi de un bit 0, extrage automat (adică şterge) bitul 0. La fel ca şi inserarea de caractere, care este complet transparentă pentru nivelul reţea din ambele calculatoare, aşa este şi inserarea de biţi. Dacă datele utilizator conţin şablonul indicator, 01111110, acest

indicator este transmis ca 011111010, dar în memoria receptorului este păstrat ca 01111110. Fig. 3-6 dă un exemplu de inserare de biţi.

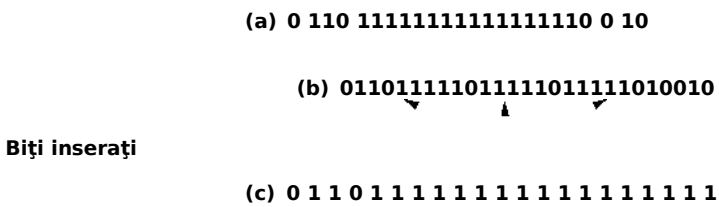


Fig. 3-6. Inserare de biţi. (a) Datele originale. (b) Datele aşa cum apar pe linie. (c) Datele aşa cum sunt stocate în memoria receptorului după extragerea biţilor inseraţi.

În cazul inserării de biţi, graniţele dintre două cadre pot fi recunoscute fără ambiguitate datorită şablonului indicator. Astfel, dacă receptorul pierde evidenţa a ceea ce primeşte, tot ceea ce are de făcut este să caute la intrare secvenţele indicator, deoarece acestea pot să apară numai la marginile cadrului şi niciodată în interiorul datelor.

Ultima metodă de încadrare este aplicabilă reţelelor în care codificarea pe mediul fizic conţine o anumită redundanţă. De exemplu, unele LAN-uri codifică un bit de date utilizând doi biţi fizici. De obicei, un bit 1 este reprezentat de o tranziţie sus-jos şi un bit 0 de o tranziţie jos-sus. Aceasta înseamnă că fiecare bit de date are o tranziţie în mijloc, receptorului fiindu-i uşor să localizeze frontierele biţilor. Combinaţiile sus-sus şi jos-jos nu sunt utilizate pentru date, dar sunt utilizate pentru delimitarea cadrelor în unele protocoale.

Ca observaţie finală asupra încadrării, multe protocoale de legătură de date utilizează, pentru o mai mare siguranţă, o combinaţie de contor de caractere cu una dintre celelalte metode. La sosirea unui cadru, pentru a localiza sfârşitul acestuia, este utilizat câmpul contor. Cadrul este acceptat ca valid doar dacă în poziţia respectivă există delimitatorul corespunzător şi dacă suma de control este corectă. Altfel, şirul de intrare este scanat pentru a detecta următorul delimitator.

3.1.3 Controlul erorilor

Problema marcării începutului şi sfârşitului fiecărui cadru fiind rezolvată, ne vom îndrepta atenţia către problema următoare: cum să ne asigurăm că toate cadrele ajung până la urmă la nivelul reţea de la destinaţie şi în ordinea corectă. Să presupunem că emiţătorul trimite cadrele de ieşire fără să verifice dacă au ajuns corect. Aşa ceva se poate accepta în cazul serviciilor neconfirmate fără conexiune, dar nu în cel al serviciilor sigure, orientate conexiune.

Modul uzual de a asigura o transmitere sigură este de a furniza emiţătorului o reacţie inversă (eng.: feedback) despre ceea ce se întâmplă la celălalt capăt al liniei. De obicei protocolul îi cere receptorului să trimită înapoi cadre de control speciale, purtând confirmări pozitive sau negative despre cadrele sosite. Dacă emiţătorul recepţionează o confirmare pozitivă despre un cadru, el ştie că acel cadru a ajuns cu bine. Pe de altă parte, o confirmare negativă înseamnă că ceva a

mers prost şi cadrul trebuie retransmis.

O complicaţie în plus vine de la posibilitatea ca defectele de echipament să determine dispariţia completă a unui cadru (de exemplu într-o rafală de zgomot). În acest caz, receptorul nu va reacţiona în nici un fel, din moment ce nu are nici un motiv să reacţioneze. Trebuie să fie clar că un protocol în care emiţătorul trimite un cadru şi apoi aşteaptă o confirmare, pozitivă sau negativă, va rămâne agăţat pentru totdeauna dacă un cadru este complet pierdut datorită, de exemplu, nefuncţio- nării echipamentului.

Această posibilitate a determinat introducerea contoarelor de timp (ceasurilor) la nivelul legăturii de date. Atunci când emiţătorul trimite un cadru, porneşte de obicei şi un contor de timp. Contorul de timp este setat să expire după un interval suficient de lung pentru ca acel cadru să poată ajunge la destinaţie, să fie prelucrat acolo şi confirmarea să se propage înapoi către emiţător. În mod normal, cadrul va fi corect recepţionat şi confirmarea va sosi înainte ca timpul să expire, caz în care contorul va fi anulat.

Dar, dacă fie cadrul, fie confirmarea se pierd, intervalul de timp expiră, în acest caz, emiţătorul fiind atenţionat că a apărut o problemă. Soluţia evidentă este retransmiterea cadrului. Dar, atunci când cadrele pot fi transmise de mai multe ori, există pericolul ca receptorul să accepte acelaşi cadru de două sau mai multe ori şi să-l trimită de mai multe ori nivelului reţea. Pentru a evita această situaţie este necesar să atribuim numere de secvenţă cadrelor de ieşire, aşa încât receptorul să poată face distincţie între cadrele retransmise şi cele originale.

Întreaga problematică a gestiunii ceasurilor şi numerelor de secvenţă, astfel încât fiecare cadru să ajungă la nivelul reţea de la destinaţie o singură dată nici mai mult, nici mai puţin, reprezintă o parte importantă a obligaţiilor nivelului legătură de date. Mai târziu, în acest capitol, urmărind o serie de exemple de complexitate din ce în ce mai mare, vom studia în detaliu cum este realizată această gestiune.

3.1.4 Controlul fluxului

Un alt aspect important de proiectare care apare la nivelul legătură de date (şi, desigur, şi la nivelurile superioare) este cum trebuie procedat cu un emiţător care doreşte în mod sistematic să transmită cadre mai repede decât poate să accepte receptorul. Această situaţie poate să apară uşor atunci când emiţătorul rulează pe un calculator rapid (sau mai puţin încărcat) şi receptorul rulează pe o maşină lentă (sau foarte încărcată). Emiţătorul continuă să transmită cadre la o rată înaltă până când receptorul este complet inundat. Chiar dacă transmisia este fără erori, la un anumit punct receptorul nu va mai fi capabil să trateze cadrele care sosesc şi va începe să piardă unele dintre ele. Bineînţeles, trebuie făcut ceva pentru a evita această situaţie.

Există două abordări des utilizate. În cazul celei dintâi, **controlul fluxului bazat pe reacţie (feedback-based flow control)**, receptorul acordă emiţătorului permisiunea de a mai transmite date, sau cel puţin comunică emiţătorului informaţii despre starea sa. În cea de-a doua, **controlul fluxului bazat pe rată (rate-based flow control)**, protocolul dispune de un mecanism integrat care

limitează rata la care emiţătorul poate transmite, fără a folosi informaţii de la receptor. În acest capitol vom studia scheme de control al fluxului bazat pe reacţie, deoarece la nivelul legătură de date nu se utilizează controlul fluxului bazat pe rată. Vom studia acest tip de control al fluxului în cap. 5.

Sunt cunoscute diferite scheme de control al fluxului, dar cele mai multe dintre ele utilizează acelaşi principiu de bază. Protocolul conţine reguli bine definite despre momentul când emiţătorul poate trimite următorul cadru. Deseori aceste reguli interzic trimiterea cadrelor înainte ca receptorul să o permită, implicit sau explicit. De exemplu, când se stabileşte o conexiune, receptorul trebuie să spună: "Acum poţi să-mi trimiţi *n* cadre, dar după ce au fost trimise, să nu trimiţi altele până când nu îţi spun să continui". Vom examina detaliile în cele ce urmează.

3.2 DETECTAREA ŞI CORECTAREA ERORILOR

Aşa cum am văzut în Cap. 2, sistemul telefonic are trei părţi: comutatoarele, trunchiurile inter- oficii (eng.: interoffice trunks) şi buclele locale. Primele două sunt acum aproape complet digitizate în majoritatea ţărilor dezvoltate. Buclele locale sunt încă din perechi de fire torsadate din cupru şi vor continua să fie aşa ani întregi, din cauza costului enorm al înlocuirii lor . În timp ce pe partea digitală erorile sunt rare, ele sunt încă obişnuite pe buclele locale. Mai mult, comunicaţia fără fir a devenit mai uzuală şi ratele erorilor sunt, în acest caz, cu câteva ordine de mărime mai proaste decât pe trunchiurile de fibră inter-oficii. Concluzia este: erorile de transmisie vor fi o realitate pentru mulţi ani de acum înainte. Trebuie găsită o metodă de tratare a acestor erori.

Ca rezultat al proceselor fizice care le generează, erorile din unele medii (de exemplu radio) tind să vină mai curând în rafale decât izolate. Sosirea erorilor în rafală are atât avantaje cât şi dezavantaje faţă de erorile izolate, de un singur bit. Avantajul este acela că datele de la calculator sunt trimise întotdeauna în blocuri de biţi. Să presupunem că dimensiunea unui bloc este de 1000 de biţi şi rata de eroare este de 0.001 per bit. Dacă erorile ar fi independente, multe blocuri ar conţine o eroare. Dacă erorile vin în rafală de câte 100, în medie vor fi afectate doar unul sau două blocuri din 100.

Dezavantajul erorilor în rafală este acela că sunt mult mai greu de detectat şi corectat decât erorile izolate.

3.2.1 Coduri corectoare de erori

Proiectanţii de reţele au dezvoltat două strategii de bază pentru tratarea erorilor. O modalitate este ca pe lângă fiecare bloc de date trimis să se includă suficientă informaţie redundantă pentru ca receptorul să poată deduce care a fost

caracterul transmis. O altă soluţie este să se includă suficientă redundanţă pentru a permite receptorului să constate că a apărut o eroare, dar nu care este eroarea, şi să ceară o retransmisie. Prima strategie utilizează **coduri corectoare de erori**, iar cea de-a doua utilizează **coduri detectoare de erori**. Folosirea codurilor corectoare de erori este deseori referită sub numele de **corectare de erori în avans (eng.: forward error correction)**.

Fiecare dintre aceste tehnici se utilizează în situaţii diferite. Pe canale cu siguranţă mare, cum ar fi fibra optică, este mai eficient să utilizăm un cod detector de erori şi să retransmitem blocul în care s-au detectat erori. În cazul canalelor de comunicaţie fără fir, este indicat sa adăugăm destulă informaţie redundantă fiecărui bloc, în loc să ne bazăm pe retransmisie, care poate să fie la rândul său afectată de erori.

Pentru a înţelege cum pot fi tratate erorile, este necesar să privim cu atenţie la ceea ce este de fapt o eroare. În mod normal, un cadru conţine m biţi de date (adică mesaj) şi r biţi redundanţi sau de control. Să considerăm lungimea totală n (adică, $n = m + r$). O unitate formată din n biţi, care conţine date şi biţi de control, este numită frecvent **cuvânt de cod** de n biţi (eng.: n-bit **codeword**).

Date fiind două cuvinte de cod, să zicem, 10001001 şi 10110001, este posibil să determinăm câţi biţi corespunzători diferă. În acest caz diferă 3 biţi. Pentru a determina câţi biţi diferă, aplicăm operatorul SAU EXCLUSIV între cele două cuvinte de cod şi numărăm biţii 1 din rezultat, de exemplu:

```
10001001
10110001
00111000
```

Numărul de poziţii binare în care două cuvinte de cod diferă se numeşte **distanţa Hamming** (Hamming, 1950). Semnificaţia sa este că dacă două cuvinte de cod sunt despărţite de o distanţă Hamming d, sunt necesare d erori de un singur bit pentru a-l converti pe unul în celălalt.

În multe aplicaţii de transmisie de date, toate cele 2^m mesaje de date posibile sunt corecte, dar, datorită modului în care sunt calculaţi biţii de control, nu sunt utilizate toate cele 2^n cuvinte de cod posibile. Dat fiind algoritmul pentru calculul biţilor de control, este posibil să construim o listă completă de cuvinte de cod permise şi din această listă să găsim cele două cuvinte de cod a căror distanţă Hamming este minimă. Această distanţă este distanţa Hamming a codului complet.

Proprietăţile detectoare şi corectoare de erori ale unui cod depind de distanţa sa Hamming. Pentru a detecta d erori, este nevoie de un cod cu distanţă $d + 1$, deoarece cu un asemenea cod nu există nici o modalitate ca d erori de un singur bit să poată modifica un cuvânt de cod corect într-un alt cuvânt de cod corect. Atunci când receptorul vede un cuvânt de cod incorect, poate spune că s-a produs o eroare de transmisie. Similar, pentru a corecta d erori, este nevoie de un cod cu distanţă $2d + 1$, deoarece în acest mod cuvintele de cod corecte sunt atât de distanţate, încât, chiar cu d modificări, cuvântul de cod originar este totuşi mai apropiat decât alte cuvinte de cod şi va fi unic determinat.

Ca un exemplu simplu de cod detector de erori, să considerăm un cod în care la

date este adăugat un singur bit de paritate. Bitul de paritate este ales astfel, încât numărul de biţi 1 din cuvântul de cod să fie par (sau impar). De exemplu, atunci când 1011010 este trimis în paritate pară, prin adăugarea unui bit la sfârşit devine 10110100. Cu paritatea impară , 1011010 devine 10110101. Un cod cu un singur bit de paritate are distanţa 2, deoarece orice eroare pe un singur bit produce un cuvânt de cod cu paritatea greşită. Acesta poate fi utilizat pentru detectarea erorilor singulare.

Ca exemplu simplu de cod corector de erori, să considerăm un cod cu numai patru cuvinte de cod corecte:

0000000000, 0000011111, 1111100000 şi 1111111111.

Acest cod are distanţa 5, ceea ce înseamnă că poate corecta erori duble. Dacă soseşte cuvântul de cod 0000000111, cel ce recepţionează ştie că originalul trebuie să fi fost 00000111111. Dacă totuşi o eroare triplă modifică 0000000000 în 0000000111, eroarea nu va fi corectată corespunzător.

Să ne imaginăm că dorim să proiectăm un cod cu m biţi de mesaj şi r biţi de control care ne va permite să corectăm toate erorile singulare. Pentru fiecare din cele 2^m mesaje corecte există n cuvinte de cod eronate, aflate la distanţă 1 de el. Acestea sunt formate prin inversarea sistematică a fiecăruia dintre cei n biţi din cuvântul de cod de n biţi format din el. Astfel, fiecare din cele 2^m mesaje corecte necesită $n + 1$ şabloane asociate. Cum numărul total de şabloane este 2^n, trebuie să avem $(n + 1)\ 2^m <= 2^n$. Utilizând $n=m+r$, această condiţie devine $(m + r + 1) <= 2^r$. Dându-se m, acesta impune o limită inferioară asupra numărului de biţi de control necesari pentru a corecta erorile singulare.

Această limită inferioară teoretică poate fi, de fapt, atinsă utilizând o metodă atribuită lui Hamming (1950). Biţii cuvântului de cod sunt numerotaţi consecutiv, începând cu bitul 1 de la marginea din stânga, bitul 2 imediat la dreapta sa, etc. Biţii care sunt puteri ale lui 2 (1, 2, 4, 8, 16 etc.) sunt biţi de control. Restul (3, 5, 6, 7, 9 etc.) sunt completaţi cu cei m biţi de date. Fiecare bit de control forţează ca paritatea unui grup de biţi, inclusiv el însuşi, să fie pară (sau impară).

Un bit poate fi inclus în mai multe calcule de paritate. Pentru a vedea la care biţi de control contribuie bitul de date din poziţia k, rescriem k ca o sumă de puteri ale lui 2. De exemplu, 11 = 1+2+8 şi 29=1+4+8+16. Un bit este verificat de acei biţi de control care apar în dezvoltarea sa (de exemplu, bitul 11 este verificat de biţii 1, 2 şi 8).

Car.	**ASCII**	**Biţi de control**
H	1001000	001100100
a	1100001	101110010
m	1101101	111010101
m	1101101	111010101
i	1101001	011010110
n	1101110	011010101
g	1100111	111110011
	0100000	100110000
c	1100011	111110000
o	1101111	001010111

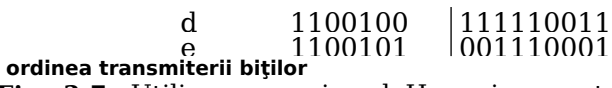


Fig. 3-7. Utilizarea unui cod Hamming pentru corectarea erorilor în rafală.

Când soseşte un cuvânt de cod, receptorul iniţializează un contor la 0. Acesta examinează apoi fiecare bit de control, k (k = 1, 2, 4, 8...) pentru a vedea dacă are paritatea corectă. Dacă nu, adaugă k la contor. Dacă, după ce au fost examinaţi toţi biţii de control, contorul este 0 (adică, dacă toţi biţii au fost corecţi), cuvântul de cod este acceptat ca valid. Dacă valoarea contorului este nenulă, ea reprezintă numărul bitului incorect. De exemplu, dacă biţii de control 1, 2 şi 8 sunt eronaţi, atunci bitul inversat este 11, deoarece este singurul verificat de biţii 1, 2 şi 8. fig. 3-6 prezintă câteva caractere ASCII pe 7 biţi codificate prin cuvinte de cod pe 11 biţi, utilizând codul Hamming. De reamintit că informaţia este regăsită în biţii de pe poziţiile 3, 5, 6, 7, 9, 10 şi 11. Codurile Hamming pot corecta numai erori singulare. Totuşi, există un artificiu care poate fi utilizat pentru a permite codurilor Hamming să corecteze erorile în rafală. O secvenţă de k cuvinte de cod consecutive este aranjată ca o matrice, având câte un cuvânt de cod pe fiecare linie. În mod normal, datele ar fi transmise linie cu linie, de la stânga la dreapta. Pentru a corecta erorile în rafală, datele vor trebui transmise pe coloane, începând cu coloana cea mai din stânga. Când au fost trimişi toţi cei k biţi, este transmisă a doua coloană şi aşa mai departe, aşa cum se arată în fig. 3-7. Atunci când un cadru ajunge la receptor, matricea este reconstruită, coloană cu coloană. Dacă a apărut o eroare în rafală, de lungime k, va fi afectat cel mult un bit din fiecare dintre cele k cuvinte de cod, dar codul Hamming poate corecta o eroare pe cuvânt de cod, aşa încât întregul bloc poate fi refăcut. Această metodă utilizează kr biţi de control pentru a face blocuri de km biţi de date imune la erorile în rafală de lungime k sau mai mică.

3.2.2 Coduri detectoare de erori

Codurile corectoare de erori sunt adesea utilizate pe canale fără fir, care sunt cunoscute ca fiind predispuse la erori, în comparaţie cu firele de cupru sau fibra optică. Fără coduri detectoare de erori, comunicaţia ar fi greu de realizat. În cazul firelor de cupru sau a fibrei optice rata erorilor este mult mai mică, aşa că detectarea erorilor şi retransmisia este de obicei mai eficientă aici ca metodă de tratare a erorilor care apar ocazional.

Ca un exemplu simplu, să considerăm un canal în care erorile sunt izolate şi rata erorilor este de 10^{-6} per bit. Să considerăm că dimensiunea unui bloc este de 1000 biţi. Pentru a permite corecţia erorilor pentru blocuri de 1000 de biţi sunt necesari 10 biţi de control; un megabit de date va necesita 10000 biţi de control. Pentru a detecta uşor un bloc cu o singură eroare de un bit, va fi suficient un bit de paritate la fiecare bloc. O dată la fiecare 1000 de blocuri va trebui transmis un extra-bloc (1001 biţi). Încărcarea suplimentară totală în cazul metodei de detecţie şi retransmisie este de numai 2001 biţi pentru un megabit de date, în comparaţie cu 10000 biţi pentru un cod Hamming.

Dacă unui bloc i se adaugă un singur bit de paritate şi blocul este puternic deformat de o eroare în rafală lungă, probabilitatea ca eroarea să fie detectată este de numai 0.5, ceea ce este greu de acceptat. Şansele pot fi îmbunătăţite considerabil dacă fiecare bloc transmis este privit ca o matrice dreptunghiulară de *n* biţi lăţime şi *k* biţi înălţime, după cum am arătat mai sus. Pentru fiecare coloană este calculat un bit de paritate, care este adăugat într-o nouă linie de la sfârşitul matricei. Matricea este apoi transmisă linie cu linie. La sosirea blocului, receptorul verifică toţi biţii de paritate. Dacă oricare din ei este greşit, va cere o retransmisie a blocului. Retransmisii succesive sunt cerute daca este nevoie, până când întregul bloc este recepţionat fără erori de paritate.

Această metodă poate detecta o singură rafală de lungime *n*, cu numai un bit pe coloană modificat. O rafală de lungime *n* + 1 va trece totuşi nedetectată dacă primul şi ultimul bit sunt inversaţi, iar toţi ceilalţi biţi sunt corecţi (o eroare în rafală nu înseamnă că toţi biţii sunt greşiţi, ci că cel puţin primul şi ultimul sunt greşiţi). Dacă blocul este puternic deformat de o rafală lungă sau de rafale scurte multiple, probabilitatea ca oricare din cele *n* coloane să aibă, accidental, paritatea corectă este 0.5, deci probabilitatea ca un bloc eronat să fie acceptat atunci când nu ar trebui este 2^{-n}.

Cu toate că schema de mai sus poate fi uneori adecvată, în practică este larg utilizată o altă metodă: **codul polinomial** (cunoscut şi sub numele de **cod cu redundanţă ciclică,** eng.: **cyclic redundancy code**). Codurile polinomiale sunt bazate pe tratarea şirurilor de biţi ca reprezentări de polinoame cu coeficienţi 0 şi 1. Un cadru de *k* biţi este văzut ca o listă de coeficienţi pentru un polinom cu *k* termeni, de la x^{k-l} la x^0. Se spune că un astfel de polinom este de gradul *k-1.* Bitul cel mai semnificativ (cel mai din stânga) este coeficientul lui x^{k-1}; următorul bit este coeficientul lui x^{k-2} ş.a.m.d. De exemplu, 110001 are şase biţi şi ei reprezintă un polinom cu şase termeni cu coeficienţii 1, 1, 0, 0, 0 şi 1:

X **U -»T⁴ l ir⁰ A t'V'ilînnmio 1 o tm •m/"\/"lTil/"\ O în nntTfnrmitofA** /TI Q1/Y/^F>TIJ/"»/^ +x +x . Aritmetica polirromială este de tip modulo 2, 111 conformitate cu regulile teoriei algebrice. Nu există transport la adunare şi nici Împrumut la scădere. Atât adunările cât şi scăderile sunt identice cu SAU EXCLUSIV. De exemplu:

10011011	00110011	11110000	01010101
+11001010	+11001101	-10100110	-10101111
01010001	11111110	01010110	11111010

Împărţirea lungă este făcută ca în binar cu excepţia faptului că scăderea este realizată modulo 2, ca mai sus. Despre un împărţitor se spune că „intră" într-un deîmpărţit dacă deîmpărţitul are tot atâţia biţi ca împărţitorul.

Atunci când este utilizată metoda codului polinomial, emiţătorul şi receptorul se pun de acord în avans asupra unui **polinom generator** *G(x).* Atât bitul cel mai semnificativ cât şi cel mai puţin semnificativ trebuie să fie 1. Pentru a calcula **suma de control** pentru un cadru cu *m* biţi, corespunzător polinomului *M(x),* cadrul trebuie să fie mai lung decât polinomul generator. Ideea este de a adăuga o sumă de control la sfârşitul cadrului, astfel încât polinomul reprezentat de cadrul cu

sumă de control să fie divizibil prin *G(x).* Când receptorul preia cadrul cu suma de control, încearcă să-l împartă la *G(x).* Dacă se obţine un rest, înseamnă că a avut loc o eroare de transmisie.

Algoritmul pentru calculul sumei de control este următorul:

1. Fie *r* gradul lui *G(x).* Se adaugă *r* biţi 0 la capătul mai puţin semnificativ al cadrului, aşa încât acesta va conţine acum *n+r* biţi şi va corespunde polinomului x'M(x).
2. Se împarte şirul de biţi ce corespund lui *G(x)* într-un şir de biţi corespunzând lui *x'M(x),* utilizând împărţirea modulo 2.
3. Se scade restul (care are întotdeauna r sau mai puţini biţi) din şirul de biţi corespunzând lui x'M(x), utilizând scăderea modulo 2. Rezultatul este cadrul cu sumă de control ce va fi transmis. Numim polinomul său *T(x).*

Fig. 3-8 ilustrează calculul pentru cadrul 1101011011 şi *G(x)* = x^4+x+1.

Ar trebui să fie clar că *T(x)* este divizibil (modulo 2) cu *G(x).* În orice problemă de împărţire, dacă din deîmpărţit se scade restul, atunci ceea ce rămâne este divizibil prin împărţitor. De exemplu, în baza 10, dacă împărţim 210278 la 10941 restul este 2399. Prin scăderea lui 2399 din 210278, ceea ce rămâne (207879) este divizibil cu 10941.

Să analizăm puterea acestei metode. Ce tipuri de erori vor fi detectate? Să ne imaginăm că apare o eroare de transmisie, aşa încât în loc să sosească şirul de biţi pentru *T(x),* ajunge *T(x) + E(X).* Fiecare bit din *E(x)* corespunde unui bit care a fost inversat. Dacă în *E(x)* există *k* biţi 1, aceasta înseamnă că au apărut *k* erori de un singur bit.

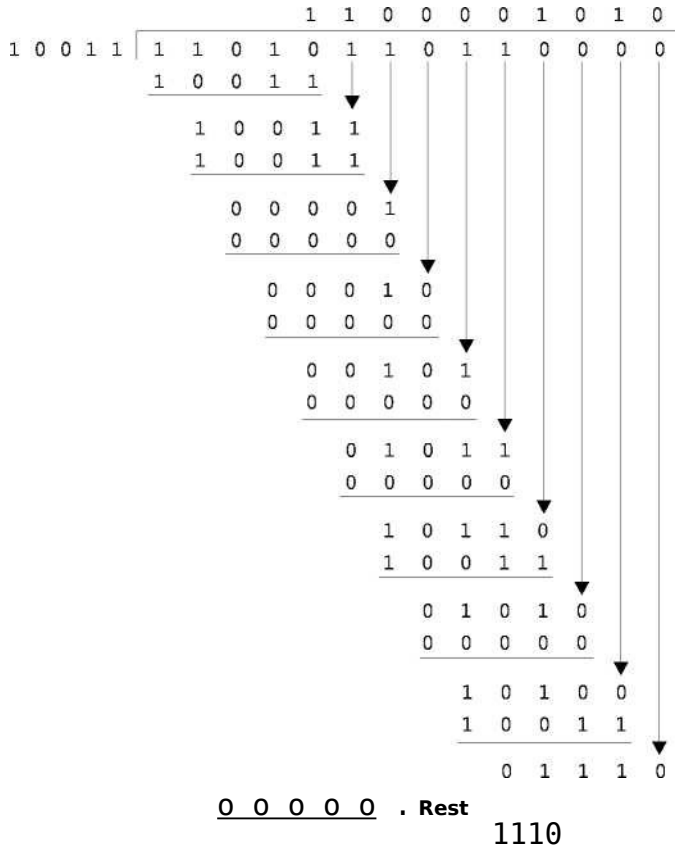
O singură eroare în rafală este caracterizată de un 1 iniţial, un amestec de 0 şi 1 şi un 1 final, toţi ceilalţi biţi fiind 0.

La recepţia cadrului cu sumă de control, receptorul îl împarte prin G(x); aceasta înseamnă că va calcula *[T(x) + E(x)]/G(x). T(x)/G(x)* este 0, aşa încât rezultatul calculului este pur şi simplu *E(x)/G(x).* Acele erori care se întâmplă să corespundă unor polinoame care îl au ca factor pe *G(x)* vor scăpa; toate celelalte vor fi detectate.

Cadru : 1101011011

Generator: 10 0 11

Mesaj după adăugarea a 4 biţi de zero: 11010110000



Cadru transmis: 11010110111110

Fig. 3-8. Calculul sumei de control în cod polinomial.

Dacă a apărut o eroare pe un singur bit, *E(x)* =x\ unde *i* determină care bit este eronat. Dacă *G(x)* conţine doi sau mai mulţi termeni, nu poate fi divizor al lui *E(x),* aşa încât toate erorile pe un singur bit vor fi detectate.

Dacă au apărut două erori izolate pe un singur bit, atunci *E(x)*= x^1+x^J, unde i>j. Alternativ, aceasta se poate scrie ca *E(x)* =$x^J(x^{i-J}+1)$. Dacă presupunem că *G(x)* nu este divizibil prin *x*, o condiţie suficientă pentru detectarea erorilor duble este ca *G(x)* să nu se dividă prin x^k+1 pentru orice *k* până la valoarea maximă *i-j* (adică, până la lungimea maximă a cadrului). Sunt cunoscute polinoame simple, de grad mic, care asigură protecţie cadrelor cu lungime mare. De exemplu, $x^{15}+x^{14}+1$ nu se va divide cu *d+1* pentru nici o valoare a lui *k* mai mică decât 32768.

Dacă există un număr impar de biţi eronaţi, *E(x)* conţine un număr impar de termeni (adică, x5+x2+1, dar nu x2+1). Interesant este că, în sistemul modulo 2 nu

există nici un polinom cu număr
impar de termeni care să îl aibă pe *x+1* ca factor. Făcându-l pe *x+1* factor al lui *G(x),* vom putea depista toate erorile constituite dintr-un număr impar de biţi inversaţi.

Pentru a demonstra că nici un polinom cu număr impar de termeni nu este divizibil cu *x+1*, să presupunem că *E(x)* are un număr impar de termeni şi este divizibil cu *x+1*. Factorizăm *E(x)* în *(x+1)Q(x).* Acum evaluăm *E(1)= (1+1)Q(1).* Deoarece 1 + 1=0 (modulo 2), *E(1)* trebuie să fie 0. Dacă *E(x)* are un număr impar de termeni, substituind fiecare *x* cu 1, rezultatul obţinut va fi întotdeauna 1. Prin urmare nici un polinom cu număr impar de termeni nu este divizibil cu *x+1*. În sfârşit, şi cel mai important, un cod polinomial cu *r* biţi de control va detecta toate erorile în rafală de lungime < *r*. O eroare în rafală de lungime *k* poate fi reprezentată de *x (x^k1 + ... +1),* unde *i* determină cât de departe este localizată rafala faţă de capătul din dreapta al cadrului recepţionat. Dacă G(x) conţine termenul x^0, atunci nu îl va avea ca factor pe *X,* aşa că dacă gradul expresiei dintre paranteze este mai mic decât gradul lui *G(x),* restul nu poate fi niciodată 0.

Dacă lungimea rafalei este *r*+1, restul împărţirii cu *G(x)* va fi zero dacă şi numai dacă rafala este identică cu *G(x).* Prin definiţia rafalei, primul şi ultimul bit trebuie să fie 1, aşa că potrivirea depinde de cei *r* – 1 biţi intermediari. Dacă toate combinaţiile sunt privite ca egal posibile, atunci probabilitatea ca un cadru incorect să fie acceptat ca valid este $1/2^{r-1}$.

Se poate arăta şi că, dacă apare o rafală de erori mai lungă de *r*+1 biţi sau dacă apar mai multe rafale mai scurte, probabilitatea ca un cadru greşit să treacă neobservat este $1/2^r$, presupunând că toate configuraţiile de biţi sunt la fel de probabile.

Anumite polinoame au devenit standarde internaţionale. Cel folosit în IEEE 802 este*:*

32 . 26 .23 .22 . 16 , 12 , 11 ,10 , 8 , 7 . 5 , 4 . 2. i , i
x “i “i “i “i *x* “i *x* “i *x* “i *x* “i *x* “i *x* “i *x* “i *x* “i *x* “i “ix1 “i I
x i *x* i *x* i *x* i *x* i *x* i *x* i *x* i *x* i *x* i *IXIXIXIXIXI.L*

Deşi calculele necesare pentru determinarea sumei control pot să pară complicate, Peterson şi Brown (1961) au arătat că pentru a calcula şi verifica suma de control poate fi utilizat un simplu registru de deplasare. În practică, acest tip de circuit este utilizat aproape întotdeauna. Este folosit aproape în toate reţelele locale, şi uneori şi în cazul liniilor punct la punct.

Timp de zeci de ani s-a presupus că acele cadre pentru care se calculează suma de control conţin biţi aleatori. Toate analizele algoritmilor pentru calculul sumei de control au fost făcute cu această presupunere. Analize mai recente ale datelor reale au arătat că această presupunere nu este corectă. Ca o consecinţă, în unele circumstanţe, erorile nedetectate sunt mult mai obişnuite decât s-a crezut anterior (Partridge ş.a. 1995).

3.3 PROTOCOALE ELEMENTARE PENTRU LEGĂTURA DE DATE

Pentru a face introducerea în subiectul protocoalelor, vom începe prin a analiza trei protocoale de complexitate din ce în ce mai mare. Pentru cititorii interesaţi, un simulator pentru aceste protocoale şi pentru cele care urmează este disponibil prin Web (vezi prefaţa). Înainte de a analiza protocoalele, este util să explicităm unele dintre ipotezele care stau la baza modelelor de comunicaţie. Pentru început, considerăm că la nivelul fizic, nivelul legătură de date şi nivelul reţea există procese independente care comunică transferându-şi mesaje în ambele sensuri. În multe cazuri, procesele de la nivelul fizic şi de legătură de date se vor executa pe un procesor dintr-un cip special de intrare/ieşire al reţelei, iar codul nivelului reţea în CPU. Sunt posibile şi alte implementări (de exemplu toate cele trei procese într-un singur cip de intrare/ieşire; nivelurile fizic şi legătură de date ca proceduri apelate de procesul nivelului reţea). În orice caz, tratarea celor trei niveluri ca procese separate va face discuţiile conceptuale mai clare şi de asemenea va scoate în evidenţă independenţa nivelurilor.

O altă presupunere cheie este că maşina *A* vrea să trimită un lung şir de date maşinii *B,* folosind un serviciu sigur, orientat pe conexiune. Mai târziu, vom considera cazul în care *B* vrea de asemenea să-i transmită simultan date lui *A*. Presupunem că *A* are tot timpul date gata de transmis şi nu aşteaptă niciodată ca aceste date să fie produse. În schimb, atunci când nivelul legătură de date al lui *A* cere date, nivelul reţea este totdeauna capabil să i le furnizeze imediat. (Şi această restricţie va fi abandonată mai târziu.)

Presupunem de asemenea că maşinile nu se defectează. Adică, aceste protocoale tratează erorile de transmisie, dar nu şi erorile generate de defectarea sau reiniţializarea calculatoarelor.

În ceea ce priveşte nivelul legătură de date, pachetul care trece de la nivelul reţea, prin interfaţă, către el este constituit din date pure, fiecare bit al acestora trebuind să fie trimis la nivelul reţea destinaţie. Faptul că nivelul reţea destinaţie poate interpreta o parte din pachetul de date ca antet nu prezintă interes pentru nivelul legătură de date.

Atunci când acceptă un pachet, nivelul legătură de date îl încapsulează într-un cadru, adăugându-i un antet şi o încheiere de legătură de date (vezi fig. 3-1). Deci un cadru se compune dintr-un pachet de date, câteva informaţii de control (antetul), şi suma de control (în încheiere) Apoi cadrul este transmis către alt nivel legătură de date. Vom presupune că există proceduri de bibliotecă adecvate pentru transmitea şi recepţionarea unui cadru: *to_physical_layer* şi, respectiv, *from_physical_layer.* Echipamentul de transmisie calculează şi adaugă suma de control (creând astfel o încheiere), astfel încât programele nivelului legătură de date nu trebuie să se preocupe de aceasta. De exemplu, ar putea fi utilizat algoritmul polinomial discutat mai devreme în acest capitol.

Iniţial, receptorul nu are nimic de făcut. Doar stă aşteptând să se întâmple ceva. În exemplele de protocoale din acest capitol arătăm că nivelul legătură de date

aşteaptă să se producă un eveniment prin apelul de procedură *wait_for_event (&event).* Această procedură redă controlul numai atunci când s-a întâmplat ceva (adică atunci când soseşte un cadru). La revenire, variabila *event* spune ce sa întâmplat. Mulţimea de evenimente posibile nu este aceeaşi pentru diferitele protocoale ce vor fi descrise şi va fi definită separat, pentru fiecare protocol în parte. De reţinut că, într-o situaţie mai realistă, nivelul legătură de date nu va sta pur şi simplu în aşteptarea unui eveniment, aşa cum am sugerat, ci va primi o întrerupere, care îl va determina să se oprească, indiferent ce făcea în acel moment, şi să se ocupe de cadrul care soseşte. Cu toate acestea,, pentru simplitate, vom ignora toate detaliile activităţilor paralele din cadrul nivelului legătură de date şi vom presupune că este tot timpul dedicat numai canalului nostru.

Când un cadru ajunge la receptor, echipamentul calculează suma de control. Dacă aceasta este incorectă (în cazul unei erori de transmisie), atunci nivelul legătură de date este informat corespunzător *(event = cksum_err).* Dacă un cadru ajunge nealterat, nivelul legătură de date este de asemenea informat *(event = frame_arrival),* aşa că poate primi cadrul pentru inspecţie folosind *from_physical_layer.* De îndată ce nivelul de legătură de date a primit un cadru nealterat, verifică informaţiile de control din antet şi dacă totul este în regulă, pachetul este transmis nivelului reţea. În nici un caz antetul nu este transmis nivelului reţea.

Există un motiv serios pentru care nivelului reţea nu trebuie să i se transmită niciodată vreo parte din antet: separarea completă a protocoalelor de reţea de cele de legătură de date. Atât timp cât nivelul reţea nu ştie nimic despre protocolul nivelului legătură de date sau despre formatul cadrului, acestea pot fi schimbate, fără să fie necesară schimbarea programelor nivelului reţea. Furnizarea unei interfeţe rigide între nivelul reţea şi nivelul legătură de date simplifică considerabil proiectarea programelor, deoarece protocoalele de comunicaţie de la niveluri diferite pot evolua independent.

```
   #define MAX_PKT 1024                  /* determină dimensiunea în octeţi a pachetului
*/
   typedef enum {false, true} boolean;                                  /*  tip  boolean
*/
   typedef unsigned int seq_nr;                            /* numere de secvenţă sau de ack
*/
   typedef struct {unsigned char data[MAX_PKT];} packet;         /*  definiţia  pachetului
*/
   typedef enum {data, ack, nak} frame_kind;             /* definiţia tipurilor de cadre
*/
   typedef struct {                             /* la acest nivel sunt transportate cadre
*/
    frame_kind kind;                                      /* ce fel de cadru este acesta?
*/
    seq_nr seq;                                                 /*  număr  de  secvenţă
*/
    seq_nr ack;                                                /*  număr  de  confirmare
*/
    packet info;                                            /* pachetul de nivel reţea
```

```
*/
   } frame;

   /* Aşteaptă producerea unui eveniment; întoarce tipul acestuia în variabila event */
   void wait_for_event(event_type *event);

   /* Preia un pachet de la nivelul reţea, spre a-l transmite prin canal */
   void from_network_layer(packet *p);

   /* Livrează nivelului reţea informaţia dintr-un cadru ajuns la destinaţie */ void
   to_network_layer(packet *p);

   /* Preia cadrul sosit de la nivelul fizic şi îl copiază în r. */
   void from_physical_layer(frame *r);

   /* Livrează cadrul nivelului fizic, pentru transmisie */
   void to_physical_layer(frame *s);

   /* Porneşte ceasul şi activează evenimentul timeout */
   void start_timer(seq_nr k);

   /* Opreşte ceasul şi dezactivează evenimentul timeout */
   void stop_timer(seq_nr k);

   /* Porneşte un ceas auxiliar şi activează evenimentul ack_timeout */
   void start_ack_timer(void);

   /* Opreşte ceasul auxiliar şi dezactivează evenimentul ack_timeout */
   void stop_ack_timer(void);

   /* Permite nivelului reţea să provoace un eveniment network_layer_ready */
   void enable_network_layer(void);

   /* Interzice nivelului reţea generarea unui eveniment network_layer_ready */
   void disable_network_layer(void);

   /* Macroinstrucţiunea inc este expandată în-line: îl incrementează circular pe k */
   #define inc(k) if (k ( MAX_SEQ) k = k + 1; else k = 0
```

Fig. 3-9. Câteva definiţii necesare în protocoalele care urmează. Aceste definiţii se găsesc în fişierul *protocol.h.*

Fig. 3-9 arată câteva declaraţii (în C) comune multor protocoale ce vor fi discutate mai târziu. Sunt definite cinci structuri de date: *boolean, seq_nr, packet, framejdnd, frame.* Un *boolean* este de tip enumerativ şi poate lua numai valorile *true* sau *false. Seq_nr* este un număr întreg mic folosit pentru a numerota cadrele, astfel încât să le putem identifica. Aceste numere de secvenţă sunt cuprinse între 0 şi *MAX_SEQ* inclusiv, aceasta din urmă fiind o constantă definită în fiecare protocol în care este necesară. Un *pachet* este unitatea de informaţie schimbată între nivelul reţea şi nivelul legătură de date de pe aceeaşi maşină sau între niveluri reţea similare. În modelul nostru el conţine totdeauna *MAX_PKT* octeţi, dar mai realist ar fi să aibă lungime variabilă.

Un *frame (cadru)* este compus din patru câmpuri: *kind, seq, ack*, şi *info*, dintre care primele trei conţin informaţii de control, iar ultimul poate conţine datele efective care trebuie transferate. Ansamblul acestor câmpuri de control este numit **antetul cadrului (frame header)**.

Câmpul *kind* (tip) spune dacă există sau nu date în cadru, deoarece unele protocoale fac distincţie între cadrele care conţin exclusiv informaţii de control şi

cele care conţin şi date. Câmpurile *seq* şi *ack* sunt utilizate pentru numere de secvenţă şi, respectiv, confirmări (*ack*nowledgements); utilizarea lor va fi descrisă în detaliu mai târziu. Câmpul *info* al unui cadru de date conţine un singur pachet de date; câmpul *info* al unui cadru de control nu este utilizat. O implementare mult mai realistă va folosi un câmp *info* de lungime variabilă, omiţându-l cu totul din cadrele de control.

Din nou, este important să ne dăm seama de relaţia dintre un pachet şi un cadru. Nivelul reţea construieşte un pachet luând un mesaj de la nivelul transport şi adăugând la acesta antetul nivelului reţea. Acest pachet este trimis nivelului legătură de date pentru a fi inclus în câmpul *info* al unui cadru care pleacă. Când cadrul ajunge la destinaţie, nivelul legătură de date extrage pachetul din cadru şi îl trimite nivelului reţea. În această manieră, nivelul reţea poate acţiona ca şi când maşinile ar putea să schimbe direct pachete.

În fig. 3-9 sunt prezentate şi câteva proceduri. Acestea sunt rutine de bibliotecă ale căror detalii sunt dependente de implementare şi al căror mod intern de lucru nu ne interesează în continuare. Procedura *wait_for_event* ciclează, aşteptând să se întâmple ceva, aşa cum am menţionat mai devreme. Procedurile *to_network_layer* şi *from_network_layer* sunt utilizate de nivelul de legătură de date pentru a trimite, respectiv a accepta, pachete de la nivelul de reţea. De reţinut că *from_physical_layer* şi *to_physical_layer* sunt utilizate pentru trimiterea cadrelor între nivelurile fizic şi legătură de date. Pe de altă parte, *to_network_layer* şi *from_network_layer* sunt folosite pentru a trimite pachetele între nivelul legătură de date şi nivelul reţea. Cu alte cuvinte, *to_network_layer* şi *from_network_layer* realizează interfaţa dintre nivelurile 2 şi 3, în timp ce *from_physical_layer* şi *to_physical_layer* realizează interfaţa dintre nivelurile 1 şi 2.

În cele mai multe dintre protocoale se consideră că se utilizează un canal nesigur care, ocazional, poate pierde cadre întregi. Pentru a contracara efectul unor asemenea calamităţi, nivelul legătură de date care transmite trebuie să pornească un contor de timp sau un ceas intern de fiecare dată când trimite un cadru. Dacă nu s-a primit un răspuns într-un interval de timp predefinit, la expirarea acestuia, nivelul legătură de date primeşte un semnal de întrerupere.

În protocoalele noastre acest lucru este asigurat de procedura *wait_for_event* care întoarce *event = timeout.* Procedurile *start_timer* şi *stop_ timer* sunt utilizate pentru a porni, respectiv a opri contorul de timp. Expirarea timpului este posibilă numai atunci când contorul de timp este pornit. Este permis explicit să se apeleze *start_ timer* în timp ce contorul de timp lucrează; un astfel de apel va reseta pur şi simplu contorul de timp, determinându-l să genereze următorul semnal de expirare de timp după ce întregul interval de timp se va epuiza (exceptând cazul în care este resetat sau dezactivat în acest interval timp).

Procedurile *start_ack_timer* şi *stop_ack_timer* sunt folosite pentru a controla un contor de timp auxiliar, utilizat pentru a genera confirmări în anumite condiţii. Procedurile *enable_network_layer* şi *disable_network_layer* sunt utilizate în protocoalele mai sofisticate, în care nu mai presupunem că nivelul de reţea are tot timpul pachete de trimis. Când nivelul legătură de date activează nivelul reţea, acestuia i se permite să întrerupă atunci când are un pachet de trimis. Indicăm

aceasta cu *event = network_layer_ready.* Când un nivel reţea este dezactivat, acesta nu poate provoca asemenea evenimente. Gestionând cu prudenţă activarea şi dezactivarea nivelului reţea, nivelul legătură de date îl poate împiedica pe acesta să-l inunde cu pachete atunci când nu mai are spaţiu în tampon.

Numerele de secvenţă ale cadrelor sunt întotdeauna de la 0 la *MAX_SEQ* (inclusiv), unde *MAX_SEQ* diferă de la protocol la protocol. Deseori este necesar ca numărul de secvenţă să avanseze circular (adică *MAX_SEQ* este urmat de 0). Această incrementare este realizată de macroinstruc- ţiunea *inc*. A fost definită ca macroinstrucţiune, deoarece este utilizată în-line în secvenţa critică. Asa cum vom vedea mai târziu, factorul care limitează performanţa reţelei este adesea prelucrarea efectuată de protocol, aşa că definirea operaţiilor simple, ca aceasta, ca macroinstrucţiuni, nu afectează claritatea codului, dar îmbunătăţeşte performanţa. Mai mult, de vreme ce *MAX_SEQ* va avea diferite valori în diferite protocoale, definirea de macroinstrucţiuni face posibilă includerea tuturor protocoalelor în acelaşi fişier binar fără conflict. Această posibilitate este utilă pentru simulator.

Declaraţiile din fig. 3-9 fac parte din fiecare dintre protocoalele care urmează. Pentru a economisi spaţiu şi pentru a furniza referinţe convenabile, acestea au fost extrase şi listate împreună, dar din punct de vedere conceptual ele trebuie incluse în protocoalele respective. În C, aceasta se realizează punând definiţiile într-un fişier antet special, în acest caz *protocol.h,* şi utilizând facilitatea #în- clude a preprocesorului C pentru a le include în fişierele protocol.

3.3.1 Un protocol simplex fără restricţii

Ca un prim exemplu vom considera cel mai simplu protocol posibil. Datele sunt transmise într-o singură direcţie. Cele două niveluri reţea, de transmisie şi de recepţie, sunt considerate tot timpul pregătite. Timpul de prelucrare poate fi ignorat. Memoria de stocare disponibilă este infinită. Şi, cel mai bun lucru dintre toate, canalul de comunicaţie între niveluri legătură de date nu pierde şi nu alterează niciodată cadrele. Acest protocol total nerealist, pe care îl vom numi "utopia", este prezentat în fig. 310. Protocolul constă din două proceduri distincte, una de emisie şi cealaltă de recepţie. Emiţătorul lucrează la nivelul legătură de date al maşinii sursă, iar receptorul la nivelul legătură de date al maşinii de destinaţie. Nu se folosesc nici numere de secvenţă, nici confirmări, aşa că nu este nevoie de *MAX_SEQ.* Singurul eveniment posibil este *frame_arrival* (sosirea unui cadru nealterat).

Emiţătorul este într-un ciclu infinit care doar inserează datele pe linie cât poate de repede. Ciclul constă din trei acţiuni: preluarea unui pachet de date de la nivelul reţea (care este întotdeauna serviabil), construirea unui cadru de ieşire folosind variabila *s* şi trimiterea cadrului pe drumul său. Acest protocol utilizează numai câmpul *info* al cadrului, deoarece celelalte câmpuri se referă la erori şi secvenţe de control, iar în cazul nostru nu există erori sau restricţii de control.

Receptorul este la fel de simplu. Iniţial el aşteaptă să se întâmple ceva, singura posibilitate fiind sosirea unui cadru nealterat. În cele din urmă, cadrul ajunge, iar procedura *wait_for_event* se întoarce cu *event* setat la *frame_arrival* (care este

oricum ignorat). Apelul rutinei *from_physical_layer* mută cadrul nou sosit din zona tampon a echipamentului în variabila *r*. În cele din urmă pachetul de date este trimis nivelului reţea şi nivelul legătură de date revine la starea de aşteptare a cadrului următor, autosuspendându-se pur şi simplu până la sosirea unui nou cadru.

```
   /* Protocolul 1 (utopia) asigură transmitere de date doar într-o direcţie, de la
   transmiţă- tor la receptor. Canalul de comunicaţie se presupune a fi fără erori, iar
   despre receptor se presupune că este capabil să prelucreze infinit de repede tot ce
   primeşte de la intrare. Deci, transmiţătorul nu face decât să stea într-o buclă,
   pompând date pe linie cât de repede poate */

   typedef enum {frame_arrival} event_type;
   #include "protocol.h"

   void sender1(void)
   {
    frame s;                                        /* tampon pentru cadrul transmis
*/
    packet buffer;                                  /* tampon pentru pachetul transmis
*/

    while (true) {
      from_network_layer(&buffer);                          /* preia ceva de transmis
*/
      s.info = buffer;                            /* îl copiază în s pentru transmitere
*/
      to_physical_layer(&s);                                /* îl trimite pe traseu */
    }
  /* Tomorrow, and tomorrow, and tomorrow

  Creeps în this petty pace from day to day

  To the last syllable of recorded time

  - Macbeth, V, v */
   }

   void receiverl(void)
   {
    frame r;
    event_type event;                   /* completat de rutina wait, dar neutilizat aici
*/

    while (true) {
      wait_for_event(&event);           /* singura posibilitate este sosirea unui cadru
*/
      from_physical_layer(&r);                          /* preia cadrul sosit
*/
      to_network_layer(&r.info);                        /* predă datele nivelului reţea */
    }
   }
```

Fig. 3-10. Un protocol simplex fără restricţii.

3.3.2 Un protocol simplu Stop-and-Wait (pas-cu-pas)

Acum vom renunţa la cea mai nerealistă restricţie utilizată în protocolul 1: posibilitatea ca nivelul reţea receptor să prelucreze datele de intrare cu viteză infinită (sau echivalent, prezenţa în nivelul legăturii de date receptor a unui tampon infinit în care să fie memorate, cât timp îşi aşteaptă rândul, toate cadrele sosite). Totuşi, se presupune în continuare că nu se produc erori pe canalul de comunicaţie şi că traficul de date este încă simplex.

Principala problemă pe care trebuie să o rezolvăm aici este cum să evităm ca emiţătorul să inunde receptorul cu date care sosesc mai rapid decât poate acesta să prelucreze. În esenţă, dacă receptorul are nevoie de un timp At ca să execute *from_physical_layer* şi *to_network_layer,* atunci emiţătorul trebuie să transmită la o viteză medie mai mică de un cadru la fiecare interval de timp de At. Mai mult, dacă presupunem că echipamentul receptor nu realizează automat memorarea în zona tam

pon şi gestiunea cozii de aşteptare, atunci emiţătorul nu trebuie să transmită niciodată un nou cadru până când cel vechi nu a fost preluat de rutina *from_physical_layer,* ca nu cumva cel nou să se scrie peste cel vechi.

În anumite situaţii speciale (de exemplu, transmisie sincronă şi un nivel legătură de date receptor complet dedicat prelucrării unei singure linii de intrare) ar putea fi posibil ca emiţătorul să introducă pur şi simplu o întârziere în protocolul 1, pentru a-l încetini suficient, astfel încât să se evite inundarea receptorului. Totuşi, de obicei, fiecare nivel legătură de date va avea mai multe linii de luat în considerare şi intervalul de timp între sosirea unui cadru şi începutul prelucrării sale poate varia considerabil. Dacă cei ce proiectează reţele pot calcula comportamentul receptorului în cazul cel mai defavorabil, atunci pot programa emiţătorul să transmită atât de încet, încât, chiar dacă fiecare cadru va suferi întârzierea maximă, nu vor exista depăşiri. Problema cu această abordare este aceea că este prea conservatoare. Ea conduce la o utilizare a lărgimii de bandă care este cu mult sub optim, cu excepţia situaţiei în care cazurile cel mai favorabil şi cel mai defavorabil sunt aproape la fel (adică, variaţia timpului de reacţie al nivelului legătură de date este mică).

```
/* Protocolul 2 (stop-and-wait) asigură la rândul său un flux de date unidirecţional,
de la emiţător la receptor. Despre canalul de comunicaţie se presupune din nou că este
fără erori, ca şi în protocolul 1. Totuşi, de data aceasta, receptorul are doar un
tampon de capacitate limitată şi o viteză de prelucrare finită, aşa că protocolul
trebuie să împiedice în mod explicit emiţătorul să inunde receptorul cu date mai repede
decât le poate trata acesta. */ typedef enum {frame_arrival} event_type; #include
"protocol.h"

void sender2(void)
{
 frame s;                                              /* tampon pentru cadrul trimis */
 packet buffer;                                      /* tampon pentru pachetul trimis */
 event_type event;                          /* singura posibilitate este sosirea unui cadru
                                                                                        */

 while                          (true)
   {     from_network_layer(&buffer);                      /* preia ceva de transmis */
   s.info          =          buffer;          /* îl copiază în s pentru transmitere */
   to_physical_layer(&s);                                  /* la revedere, micuţ cadru */
   wait_for_event(&event);                      /* nu continuă până nu primeşte semnalul */
 }
}

void receiver2(void)
{                                                        /* zone tampon pentru cadre */
 frame r, s;                               /* singura posibilitate este sosirea unui cadru
 event_type event;                                                                      */

 while (true) {
   wait_for_event(&event);              /* singura posibilitate este sosirea unui cadru
*/
   from_physical_layer(&r);                                    /*  preia  cadrul  sosit
*/
   to_network_layer(&r.info);                        /* livrează datele nivelului reţea
*/
```

```
      to_physical_layer(&s);          /* trimite un cadru fictiv pentru a trezi emiţătorul
*/
    }
   }
```

Fig. 3-11. Un protocol simplex stop-and-wait.

O soluţie mult mai generală a acestei dileme este ca receptorul să furnizeze o reacţie către emiţător. După trimiterea unui pachet către nivelul său reţea, receptorul trimite un mic cadru fictiv către emiţător care, de fapt, îi dă emiţătorului permisiunea să transmită următorul cadru. După ce a transmis un cadru, emiţătorul este obligat de protocol să intre în aşteptare un timp, până când soseşte micul cadru fictiv (deci confirmarea). Utilizarea reacţiei de la receptor pentru a anunţa emiţătorul că poate trimite date este un exemplu de control al fluxului menţionat anterior.

Protocoalele în care emiţătorul trimite un cadru şi apoi, înainte de a continua, aşteaptă o confirmare, se numesc **stop-and-wait** (pas-cu-pas). Fig. 3-11 prezintă un exemplu de protocol simplex *stop- and-wait.* Chiar dacă traficul de date este simplex, mergând numai de la emiţător la receptor, cadrele se deplasează în ambele direcţii. În consecinţă, canalul de comunicaţie dintre cele două niveluri legătură de date trebuie să permită transferul de informaţie bidirecţional. Totuşi, acest protocol impune o alternanţă strictă a fluxului: mai întâi emiţătorul trimite un cadru, apoi receptorul trimite un alt cadru, apoi emiţătorul trimite alt cadru ş.a.m.d. În acest caz este suficient un canal fizic semiduplex.

Ca şi în protocolul 1, emiţătorul începe prin preluarea unui pachet de la nivelul reţea, utilizarea lui pentru construirea unui cadru şi trimiterea acestuia. Numai că acum, spre deosebire de protocolul 1, emiţătorul trebuie să aştepte până când soseşte un cadru de confirmare, înainte de a relua ciclul şi a prelua următorul pachet de la nivelul reţea. Nivelul legătură de date care transmite nu are nevoie să inspecteze cadrul care soseşte: nu există decât o singură posibilitate. Cadrul primit este întotdeauna o confirmare.

Singura diferenţă dintre *receiverl* şi *receiver2* este aceea că după transmiterea unui pachet către nivelul reţea, *receiver2* trimite un cadru de confirmare înapoi la emiţător, înainte de a intra din nou în bucla de aşteptare. Deoarece numai sosirea cadrului de întoarcere la emiţător este importantă, nu şi conţinutul lui, receptorul nu trebuie să pună nici o informaţie particulară în el.

3.3.3 Un protocol simplex pentru un canal cu zgomote

Să considerăm situaţia normală a unui canal de comunicaţie care pot apărea erori. Cadrele pot fi fie modificate, fie complet pierdute. Totuşi, presupunem că dacă un cadru a fost modificat în tranzit, echipamentul receptor va detecta acest lucru atunci când calculează suma de control. Dacă un cadru este modificat într-un asemenea mod, încât suma de control este totuşi corectă, situaţie care este foarte puţin probabilă, acest protocol (şi toate celelalte protocoale) pot eşua (adică, trimit un pachet incorect către nivelul reţea).

La prima vedere s-ar părea că o variantă a protocolului 2 va funcţiona: adăugarea unui contor de timp (ceas). Emiţătorul poate trimite un cadru, dar

receptorul va trimite un cadru de confirmare numai dacă informaţia a fost recepţionată corect. Dacă la receptor ajunge un cadru modificat, el va fi eliminat. După un timp, emiţătorul va ieşi din aşteptare şi va retrimite cadrul. Acest proces va fi repetat până când cadrul va ajunge în final intact. Schema de mai sus conţine o eroare fatală. Gân- diţi-vă la problemă şi încercaţi să descoperiţi ce este greşit înainte să citiţi mai departe.

Pentru a vedea ce poate merge rău, amintiţi-vă care este sarcina proceselor nivelului legătură de date – aceea de a asigura comunicaţie fără erori, transparentă, între procesele nivelului reţea. Nivelul reţea de pe maşina *A* dă o serie de pachete nivelului său legătură de date, care trebuie să asigure o serie identică de pachete nivelului reţea de pe maşina *B* prin nivelul său legătură de date. În particular, nivelul reţea de pe *B* nu are nici o posibilitate să ştie că un pachet a fost pierdut sau duplicat, aşa că nivelul legătură de date trebuie să garanteze că nici o combinaţie de erori de transmisie, indiferent cât de puţin probabile, nu poate produce un pachet duplicat care să fie transmis nivelului reţea.

Să considerăm următorul scenariu:

1. Nivelul reţea de pe *A* trimite pachetul 1 către nivelul său legătură de date. Pachetul este corect recepţionat de *B* şi este trimis nivelului reţea de pe *B. B* trimite un cadru de confirmare înapoi lui *A*.
2. Cadrul de confirmare s-a pierdut complet. El nu va mai ajunge deloc. Viaţa ar fi cu mult mai simplă în cazul în care canalul ar altera sau pierde doar cadre de date, nu şi cadre de control, dar, din nefericire, canalul nu face discriminări.
3. Nivelul de legătură de date de pe *A* aşteaptă expirarea timpului limită. Nerecepţionând o confirmare, el presupune (incorect) că acel cadru de date a fost modificat sau pierdut şi trimite încă o dată cadrul conţinând pachetul 1.
4. Cadrul duplicat ajunge şi el cu bine la nivelul legătură de date *B* şi este trimis nivelului reţea de acolo. Dacă *A* trimite un fişier lui *B*, o porţiune de fişier va fi duplicată (adică, copia fişierului făcută de *B* va fi incorectă şi eroarea nu va fi detectată). Cu alte cuvinte, protocolul va eşua.

În mod clar, este necesară o soluţie ca receptorul să poată distinge un cadru pe care îl vede pentru prima dată de o retransmisie. Soluţia evidentă este aceea ca emiţătorul să pună un număr de secvenţă în antetul fiecărui cadru pe care îl trimite. Apoi receptorul poate verifica numărul de secvenţă al fiecărui cadru sosit pentru a vedea dacă este un cadru nou sau un duplicat ce trebuie eliminat.

Deoarece este de dorit ca un antet de cadru să fie de dimensiune mică, se pune întrebarea următoare: care este numărul minim de biţi necesari pentru numărul de secvenţă? Singura ambiguitate în acest protocol este între un cadru m şi succesorul său m+1. În cazul în care cadrul m este pierdut sau modificat, receptorul nu îl va confirma, aşa încât emiţătorul va încerca să-l retransmită. Odată ce a fost corect recepţionat, receptorul va trimite o confirmare înapoi la emiţător. Aici este punctul în care putem să avem necazuri. După cum cadrul de confirmare ajunge sau nu corect înapoi la emiţător, emiţătorul va încerca să transmită m sau m+1.

Evenimentul care determină emiţătorul să înceapă transmiterea lui *m*+2 este sosirea unei confirmări pentru *m*+1. Dar aceasta presupune că *m* a fost recepţionat corect şi, mai mult, confirmarea a fost de asemenea corect recepţionată de emiţător (altfel emiţătorul nu ar fi trimis *m*+1, ca să nu mai vorbim de *m*+2). În consecinţă, singura ambiguitate este între un cadru şi predecesorul sau succesorul său imediat, nu între ultimii doi.

Este deci suficient un număr de secvenţă de 1 bit (0 sau 1). La fiecare moment de timp, receptorul aşteaptă un anumit număr de secvenţă. Orice cadru sosit, care conţine un număr de secvenţă greşit este rejectat ca duplicat. Atunci când soseşte un cadru cu număr de secvenţă corect, acesta este acceptat şi transmis nivelului reţea. Apoi numărul de secvenţă aşteptat este incrementat modulo 2 (adică, 0 devine 1 şi 1 devine 0).

Un exemplu de astfel de protocol este prezentat în fig. 3-12. Protocoalele în care emiţătorul aşteaptă pentru o confirmare pozitivă înaintea de a trece la următorul element de date se numesc deseori **PAR (Positive Acknowledgement with Retransmisión**, rom.: confirmare pozitivă cu retransmitere) sau **ARQ** (**Automatic Repeat reQuest**, rom: cerere automată de repetare). Asemenea protocolului 2, şi acesta transmite datele într-o singură direcţie.

Protocolul 3 se deosebeşte de predecesorii săi prin aceea că şi emiţătorul şi receptorul au o variabilă a cărei valoare este păstrată cât timp nivelul legătură de date este în starea de aşteptare. Emiţătorul păstrează numărul de secvenţă al următorului cadru de transmis în *nextjrame_to_send;* receptorul păstrează numărul de secvenţă al următorului cadru aşteptat în *frame_expected.* Fiecare protocol are o scurtă fază de iniţializare înainte de a intra în bucla infinită.

```
/* Protocolul 3 (par) permite un flux de date unidirecţional, printr-un canal nesigur. */

#define MAX_SEQ 1                           /* trebuie să fie 1 pentru protocolul 3 */
typedef enum {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"

void sender3(void)
{
  seq_nr next_frame_to_send;        /* numărul de secvenţă al următorului cadru trimis */
  frame s;                                                  /* variabilă temporară */
  packet buffer;                                  /* tampon pentru pachetul transmis */
  event_type event;

  next_frame_to_send = 0;              /* iniţializează numerele de secvenţă de ieşire */
  from_network_layer(&buffer);                              /* preia primul pachet */
  while (true) {
    s.info = buffer;                        /* construieşte un cadru pentru transmitere */
    s.seq = next_frame_to_send; /* inserează în cadru un număr de secvenţă */
    to_physical_layer(&s);                                  /* îl trimite pe traseu */
    start_timer(s.seq);         /* dacă răspunsul întârzie prea mult, timpul va expira */
    wait_for_event(&event);                       /* frame_arrival, cksum_err, timeout */
    if (event == frame_arrival) {                           /* preia confirmarea */
      from_physical_layer(&s);
     if(s.ack==next_frame_to_send) {
       stop_timer(s.ack);                                          /* opreşte ceasul */
       from_network_layer(&buffer);               /* preia următorul pachet de transmis */
       inc(next_frame_to_send);                      /* inversează next_frame_to_send */
    }
  }
  }
}

void receiver3(void)
{
  seq_nr frame_expected;
  frame r, s;
  event_type event;

  frame_expected = 0; while (true) { wait_for_event(&event); if (event == frame_arrival) {
  from_physical_layer((r);
      if (r.seq == frame_expected)             /* variante: frame_arrival, cksum_err */
        { to_network_layer((r.info);                    /* a sosit un cadru corect */ /*
        inc(frame_expected);                           preia cadrul nou sosit */ /* este
      }                                                    cel pe care îl aşteptam */ /*
      s.ack=1-frame_expected;                           transferă datele nivelului reţea
      to_physical_l ayer(&s);                               */ /* incrementează număr de
    }                                                                    secvenţă */
  }
}                                           /* spune care frame este confirmat */ /*
                                        nici unul dintre câmpuri nu este utilizat */
```

Fig. 3-12. Un protocol cu confirmare pozitivă şi retransmitere.

După transmiterea unui cadru, emiţătorul declanşează contorul de timp. Dacă acesta era deja pornit, atunci va fi resetat pentru un nou interval complet. Intervalul de timp trebuie să fie ales astfel, încât să permită sosirea cadrului la receptor, prelucrarea sa de către receptor, chiar şi în cazul cel mai defavorabil, şi

propagarea cadrului de confirmare înapoi la emiţător. Numai atunci când a expirat acest interval de timp se poate spune cu siguranţă că s-a pierdut fie cadrul transmis, fie confirmarea sa şi se poate trimite un duplicat. Dacă intervalul de timp este prea mic, emiţătorul va transmite cadre care nu sunt necesare. Aceste cadre nu influenţează corectitudinea protocolului, dar îi afectează performanţele.

După transmiterea unui cadru şi pornirea contorului de timp, emiţătorul aşteaptă să se întâmple ceva interesant. Există doar trei posibilităţi: un cadru de confirmare ajunge intact, soseşte un cadru de confirmare eronat sau expiră timpul. Dacă soseşte o confirmare validă, atunci emiţătorul preia următorul pachet de la nivelul reţea şi îl pune în tampon, scriind peste pachetul anterior. De asemenea avansează numărul de secvenţă. Dacă soseşte un cadru modificat sau nu soseşte nici un cadru, nu se modifică nici tamponul şi nici numărul de secvenţă, aşa că poate fi transmis un duplicat.

Atunci când la receptor soseşte un cadru corect, este verificat numărul de secvenţă, pentru a vedea dacă nu cumva este un duplicat. Dacă nu, este acceptat, transmis nivelului reţea şi este generată o confirmare. Cadrele duplicate şi cele modificate nu sunt trimise către nivelul reţea.

3.4 PROTOCOALE CU FEREASTRĂ GLISANTĂ

În protocoalele anterioare, cadrele cu date erau transmise într-o singură direcţie. În cele mai multe situaţii practice, este necesar să se transmită date în ambele direcţii. O modalitate de a realiza transmisia de date full-duplex este de a avea două canale de comunicaţie separate, fiecare dintre ele fiind utilizat pentru traficul de date simplex (în direcţii diferite). Dacă se face aceasta, vom avea două circuite fizice separate, fiecare cu un canal "direct" (eng.: forward) pentru date şi un canal "invers" (eng.: reverse) pentru confirmări. În ambele cazuri lărgimea de bandă a canalului invers este irosită aproape în totalitate. Ca efect, utilizatorul plăteşte pentru două circuite, dar utilizează doar capacitatea unuia.

O idee mai bună este să se utilizeze acelaşi circuit pentru date în ambele direcţii. În definitiv, la protocoalele 2 şi 3 a fost deja utilizată transmiterea cadrelor în ambele sensuri şi canalul invers are aceeaşi capacitate ca şi canalul direct. În acest model, cadrele cu date de la *A* la *B* sunt amestecate cu cadrele de confirmare de la *A* la *B*. Uitându-se la câmpul *kind* din antetul cadrului ce a sosit, receptorul poate spune dacă este vorba de un cadru de date sau de confirmare.

Cu toate că întrepătrunderea cadrelor de date şi control pe acelaşi circuit constituie o îmbunătăţire faţă de cazul utilizării a două circuite fizice separate, mai este posibilă încă o îmbunătăţire. Atunci când soseşte un cadru cu date, în locul emiterii imediate a unui cadru de control separat, receptorul stă şi aşteaptă până când nivelul reţea îi dă următorul pachet. Confirmarea este ataşată cadrului cu date de ieşire (utilizând câmpul *ack* din antetul cadrului). De fapt, confirmarea este

transportată pe gratis de către următorul cadru cu date de ieşire. Tehnica întârzierii confirmării, astfel încât să poată fi agăţată de următorul cadru de date, este cunoscută ca **ataşare** (eng.: **piggybacking**).

Principalul avantaj al utilizării tehnicii de ataşare în comparaţie cu utilizarea cadrelor de confirmare distincte este o mai bună utilizare a lărgimii de bandă disponibile. Câmpul *ack* din antetul cadrului ocupă doar câţiva biţi, în timp ce un cadru separat va necesita un antet, confirmarea şi o sumă de control. În plus, mai puţine cadre transmise înseamnă mai puţine întreruperi datorate "sosirii cadrelor" şi probabil mai puţine zone tampon în receptor, în funcţie de modul în care sunt organizate programele receptorului. În următorul protocol ce va fi examinat, câmpul de ataşare ocupă doar un bit în antetul cadrului. Arareori ocupă mai mult de câţiva biţi.

Totuşi, tehnica de ataşare introduce o complicaţie care nu era prezentă în cazul confirmărilor separate. Cât timp trebuie să aştepte nivelul legătură de date pachetul la care să ataşeze confirmarea? Dacă nivelul legătură de date aşteaptă mai mult timp decât perioada de timeout a emiţătorului, cadrul va fi retransmis, anulând complet rolul confirmărilor. Dacă nivelul legătură de date ar fi un oracol şi ar putea prezice viitorul, ar putea şti când va sosi următorul pachet de la nivelul reţea şi ar putea decide dacă să îl aştepte sau să trimită imediat o confirmare separată, în funcţie de cât de lungă urmează să fie aşteptarea. Desigur, nivelul legătură de date nu poate prezice viitorul, aşa că trebuie să recurgem la câteva scheme ad-hoc, cum ar fi aşteptarea pentru un număr fixat de milisecunde. Dacă un nou pachet soseşte repede, confirmarea este adăugată în el; altfel, dacă până la sfârşitul acestei perioade de timp nu a sosit un nou pachet, nivelul legătură de date trimite un cadru de confirmare separat.

Următoarele trei protocoale sunt protocoale bidirecţionale care aparţin unei clase de protocoale numite protocoale cu **fereastră glisantă** (eng.: **sliding window**). Cele trei diferă între ele în termeni de eficienţă, complexitate şi necesar de tampoane, aşa cum vom vedea mai târziu. În cadrul acestora, ca în toate protocoalele cu fereastră glisantă, fiecare cadru expediat conţine un număr de secvenţă cuprins între 0 şi o valoare maximă. Maximul este de obicei 2^n -l, ca numărul de secvenţă să se încadreze exact într-un câmp de n biţi. Protocoalele cu fereastră glisantă pas-cu-pas utilizează $n = 1$, restricţionând numerele de secvenţă la 0 şi 1, dar versiuni mai sofisticate pot utiliza o valoare arbitrară a lui n.

Esenţa protocoalelor cu fereastră glisantă este aceea că, la orice moment de timp, emiţătorul menţine o mulţime de numere de secvenţă care corespund cadrelor pe care are permisiunea să le trimită. Se spune că aceste cadre aparţin **ferestrei de transmisie** (eng.: **sending window**). Similar, receptorul menţine de asemenea o **fereastră de recepţie** (eng.: **receiving window**), ce corespunde mulţimii de cadre care pot fi acceptate. Fereastra emiţătorului şi fereastra receptorului nu trebuie să aibă aceleaşi limite minime şi maxime şi nici măcar aceeaşi dimensiune. În unele protocoale ele au dimensiune fixă, dar în altele ele pot creşte sau scădea pe măsură ce cadrele sunt emise sau recepţionate.

Chiar dacă aceste protocoale dau nivelului legătură de date mai multă independenţă în ceea ce priveşte ordinea în care poate primi sau recepţiona cadre,

nu am renunţat la cerinţa ca protocolul să livreze pachetele la nivelul reţea destinaţie în aceeaşi ordine în care acestea sunt trimise către nivelul legătură de date de pe maşina emiţătoare. Nu am modificat nici condiţia impusă canalului fizic de comunicaţie, care trebuie să se comporte "ca un fir", adică trebuie să trimită toate cadrele în ordinea emiterii.

Numerele de secvenţă din cadrul ferestrei emiţătorului reprezintă cadre transmise sau cadre ce pot fi transmise, dar încă neconfirmate. De fiecare dată când de la nivelul reţea soseşte un nou pachet, acestuia îi este atribuit următorul număr de secvenţă, iar marginea superioară a ferestrei este avansată cu unu. Atunci când soseşte o confirmare, creşte cu unu limita inferioară a ferestrei. În acest mod, fereastra menţine continuu o listă de cadre neconfirmate. Un exemplu este prezentat în fig. 3-13.

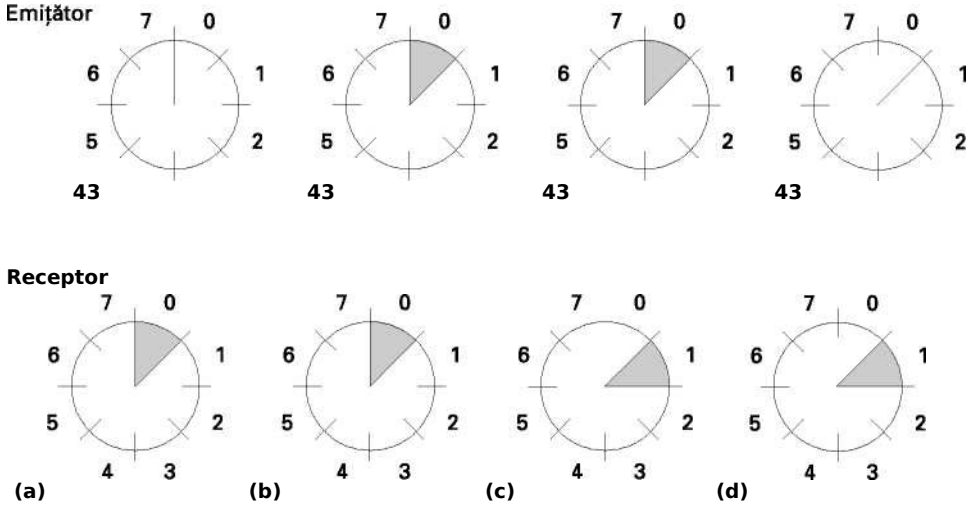


Fig. 3-13. O fereastră glisantă de dimensiune 1, cu număr de secvenţă de 3 biţi. (a) Iniţial. (b) După ce a fost transmis primul cadru. (c) După ce a fost recepţionat primul cadru. (d) După ce a fost recepţionată prima confirmare.

Deoarece cadrele din fereastra curentă a emiţătorului pot fi pierdute sau modificate pe traseu, emiţătorul trebuie să păstreze toate aceste cadre în memoria sa pentru o posibilă retransmisie. Astfel, dacă dimensiunea maximă a ferestrei este *n,* emiţătorul are nevoie de *n* tampoane pentru a păstra cadrele neconfirmate. Dacă fereastra creşte la dimensiunea maximă, nivelul legătură de date al emiţătorului trebuie să forţeze închiderea nivelului reţea până când se eliberează un tampon.

Fereastra nivelului legătură de date receptor corespunde cadrelor pe care acesta le poate accepta. Orice cadru din afara ferestrei este eliminat fără comentarii. Atunci când este recepţionat un cadru al cărui număr de secvenţă este egal cu marginea inferioară a ferestrei, acesta este trimis nivelului reţea, este generată o confirmare şi fereastra se deplasează cu o unitate. Spre deosebire de fereastra emiţătorului, fereastra receptorului rămâne întotdeauna la dimensiunea iniţială. De notat că o fereastră de dimensiune 1 înseamnă că nivelul legătură de date acceptă numai cadre ordonate, dar pentru ferestre mari afirmaţia nu mai este valabilă. Nivelul reţea este, dimpotrivă, alimentat întotdeauna cu date în ordine

corectă, indiferent de dimensiunea ferestrei nivelului legătură de date.

Fig. 3-13 prezintă un exemplu cu o fereastră de dimensiune maximă 1. Iniţial, nu sunt emise cadre, aşa că marginile inferioară şi superioară ale ferestrei emiţătorului sunt egale, dar o dată cu trecerea timpului, situaţia evoluează ca în figură.

3.4.1 Un protocol cu fereastră glisantă de un bit

Înainte de a analiza cazul general, să examinăm mai întâi un protocol cu fereastră glisantă având dimensiunea maximă a ferestrei 1. Un astfel de protocol utilizează metoda stop-and-wait, deoarece emiţătorul transmite un cadru şi aşteaptă confirmarea sa înaintea transmiterii următorului cadru.

Fig. 3-14 prezintă un astfel de protocol. Ca şi alte protocoale, acesta începe prin definirea unor variabile. *Nextjrame_ţo_send* arată ce cadru încearcă să transmită emiţătorul. Similar, *frame_ expected* arată ce cadru este aşteptat de receptor. În ambele cazuri singurele posibilităţi sunt 0 şi 1.

```
/* Protocolul 4 (fereastră glisantă) este bidirecţional. */

#define MAX_SEQ 1                              /* pentru protocolul 4 trebuie să fie 1 */
typedef enum {frame_arrival, cksum_err, timeout} event_type;
#include "protocol.h"


 void protocol4(void)
 {                                                                    /*  doar  0  sau  1
 seq_nr next_frame_to_send;                                        */ /* doar 0 sau 1 */ /*
 seq_nr frame_expected; frame r, s;                                   variabile temporare */
 packet buffer; event_type event;                 /* pachetul curent, care este transmis */

 next_frame_to_send = 0;
 frame_expected = 0;                                  /* următorul cadru pe fluxul de ieşire
 from_network_layer(&buffer); s.info               */ /* numărul de secvenţă al cadrului aşteptat
 = buffer;                                         */
 s.seq = next_frame_to_send;                          /* preia un pachet de la nivelul reţea
 s.ack = 1 − frame_expected;                       */ /* pregăteşte trimiterea cadrului iniţial
 to_physical_layer(&s);                            */ /* inserează în cadru numărul de secvenţă
 start_timer(s.seq);                               */  /*  confirmare  ataşată  */  /*  transmite
 while (true)                                      cadrul */ /* porneşte ceasul */
 { wait_for_event(&event); if (event
 == frame_arrival)                                    /* frame_arrival, cksum_err sau timeout */
 { from_physical_layer(&r); if                     /* un cadru a ajuns nedeformat */ /* preia
 (r.seq == frame_expected)                         cadrul */ /* tratează fluxul de cadre sosite
 { to_network_l ayer(&r.info);                     */ /* predă pachetul nivelului reţea */ /*
 inc(frame_expected);                              inversează numărul de secvenţă aşteptat */
      }
      if (r.ack ==                                    { /* tratează fluxul de cadre de ieşire */
        next_frame_to_send)                    /* opreşte ceasul */ /* preia un nou pachet de la
        stop_timer(r.ack);                     nivelul reţea */ /*inversează numărul de secvenţă al
        from_network_layer(&buffer);           emiţătorului */
        i nc(next_frame_to_send);
      }
    }                                          /* construieşte cadrul care iese */ /* inserează un
                                               număr de secvenţă */ /* numărul de secvenţă al
                                               ultimului cadru primit */ /* transmite un cadru
                                                                          */ /* porneşte ceasul */
    s.info = buffer;
    s.seq = next_frame_to_send;
    s.ack = 1 − frame_expected;
    to_physical_layer(&s);
    start_timer(s.seq);
 }
 }
```

Fig. 3-14.Un protocol cu fereastră glisantă de 1 bit.

În mod normal, unul dintre cele două niveluri legătură de date porneşte primul trimiţând primul cadru. Cu alte cuvinte, numai unul din programele nivelului legătură de date va conţine apelurile procedurilor *to_physical_layer* şi *start_timer* în afara buclei principale. În eventualitatea că ambele niveluri legătură de date pornesc simultan, apare o situaţie specială, care va fi discutată mai târziu. Maşina care porneşte prima preia primul pachet de la nivelul reţea propriu, construieşte din el un cadru şi îl trimite. Când acest cadru (sau oricare altul) soseşte, nivelul legătură de date receptor verifică dacă nu cumva este un duplicat, exact ca în protocolul 3. Dacă respectivul cadru este cel aşteptat, atunci este

trimis nivelului reţea şi fereastra receptorului este deplasată.

Câmpul de confirmare conţine numărul ultimului cadru recepţionat fără eroare. Dacă acest număr corespunde cu numărul de secvenţă al cadrului pe care emiţătorul încearcă să-l transmită, emiţătorul ştie că a terminat cu cadrul memorat în tampon şi poate prelua următorul pachet de la nivelul său reţea. Dacă numărul de secvenţă nu corespunde, el trebuie să continue să trimită acelaşi cadru. De fiecare dată când este recepţionat un cadru, un alt cadru este trimis de asemenea înapoi.

Să examinăm acum cât de fiabil este protocolul 4 la condiţiile limită. Să presupunem că *A* încearcă să trimită cadrul 0 lui *B* şi *B* încearcă să trimită cadrul său 0 lui *A*. Să presupunem că *A* trimite un cadru lui *B*, dar timpul de expirare al lui *A* este puţin prea scurt. În consecinţă, datorită expirării repetate a timpului, *A* va trimite o serie de cadre identice, toate cu *seq=0* şi *ack*= 1.

Atunci când la *B* soseşte primul cadru corect, el va fi acceptat şi *frame_expected* va fi setat la 1. Toate cadrele următoare vor fi respinse, deoarece B aşteaptă cadre cu numărul de secvenţă 1, nu 0. Mai mult, deoarece toate duplicatele au *ack*= 1 şi B este încă în aşteptarea confirmării lui 0, B nu va prelua un nou pachet de la nivelul său reţea.

După sosirea fiecărui duplicat respins, B trimite lui A un cadru conţinând *seq=0*

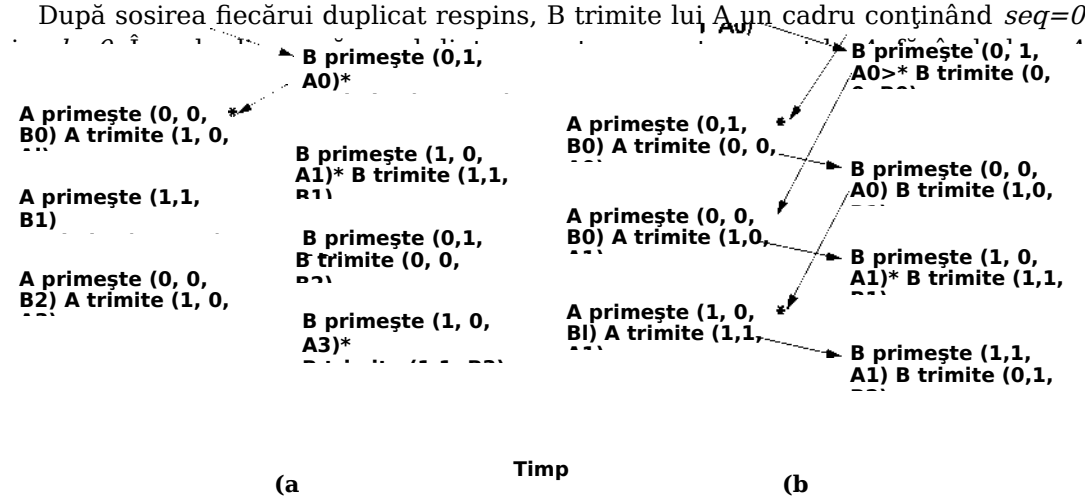

B primeşte (0,1, A0)*
A primeşte (0, 0, B0) A trimite (1, 0,
A primeşte (1,1, B1)
B primeşte (1, 0, A1)* B trimite (1,1,
B primeşte (0,1, B trimite (0, 0,
A primeşte (0, 0, B2) A trimite (1, 0,
B primeşte (1, 0, A3)*
A primeşte (0,1, B0) A trimite (0, 0,
B primeşte (0, 1, A0>* B trimite (0,
B primeşte (0, 0, A0) B trimite (1,0,
A primeşte (0, 0, B0) A trimite (1,0,
B primeşte (1, 0, A1)* B trimite (1,1,
A primeşte (1, 0, Bl) A trimite (1,1,
B primeşte (1,1, A1) B trimite (0,1,
Timp
(a
(b

Fig. 3-15. Două scenarii pentru protocolul 4. (a) Cazul normal. (b) Cazul special. Notaţia este (seq, ack, packet number). Un asterisc arată că nivelul reţea acceptă un pachet.

Totuşi, dacă ambele părţi trimit simultan un pachet iniţial, atunci apare o situaţie specială. Această dificultate de sincronizare este ilustrată de fig. 3-15. În partea (a) este prezentată funcţionarea normală a protocolului. În (b) este ilustrată situaţia specială. Dacă *B* aşteaptă primul cadru de la *A* înainte de a-l trimite pe al său, secvenţa de acţiuni este cea arătată în (a) şi fiecare cadru este acceptat. Totuşi, dacă *A* şi *B* iniţiază comunicaţia simultan, primele lor cadre se încrucişează şi nivelurile legătură de date ajung în situaţia (b). În (a) fiecare sosire de cadru aduce un nou pachet pentru nivelul reţea; nu există duplicate. În (b) jumătate din cadre conţin duplicate, chiar dacă nu există erori de transmisie. Situaţii similare pot

să apară ca rezultat al expirării premature a timpului, chiar dacă una dintre părţi începe prima. De fapt, dacă intervin mai multe expirări premature, atunci cadrele pot fi trimise de trei sau mai multe ori.

3.4.2 Un protocol de revenire cu *n* paşi (Go Back n)

Până acum am făcut presupunerea tacită că timpul de transmisie necesar pentru ca un cadru să ajungă la receptor plus timpul de transmisie a confirmării este neglijabil. Uneori această presupunere este în mod cert falsă. În aceste situaţii timpul mare de transfer poate avea implicaţii importante pentru eficienţa utilizării lărgimii de bandă. Ca exemplu, să considerăm un canal de satelit de 50 Kbps cu timpul de întârziere datorită propagării dus-întors de 500 milisecunde. Să ne imaginăm că încercăm să utilizăm protocolul 4 pentru a trimite cadre de 1000 de biţi prin satelit. La $t = 0$ emiţătorul începe să trimită primul cadru. Considerând cele mai optimiste condiţii (fără aşteptare la receptor şi un cadru de confirmare scurt), cadrul nu poate ajunge în totalitate la receptor înainte de $t = 270$ milisecunde, iar confirmarea nu poate ajunge înapoi la emiţător înainte de $t = 520$ milisecunde. Aceasta înseamnă că emiţătorul a fost blocat pentru 500/520 sau 96% din timp. Cu alte cuvinte, a fost utilizată doar 4% din lărgimea de bandă. Evident, combinaţia dintre un timp de tranziţie lung, lărgime de bandă mare şi un cadru de lungime mică este dezastruoasă din punct de vedere al eficienţei.

Problema descrisă anterior poate fi privită ca o consecinţă a regulii care cere ca un emiţător să aştepte o confirmare înaintea trimiterii unui alt cadru. Dacă relaxăm această restricţie, poate fi atinsă o eficienţă mult mai ridicată. Practic, soluţia constă în a permite emiţătorului să transmită până la *w* cadre, în loc de unul singur. Cu o alegere potrivită a lui *w* emiţătorul va putea să transmită continuu cadre pentru un timp egal cu timpul de tranzit, fără a umple fereastra. În exemplul anterior *w* va fi minim 26. Emiţătorul începe emiterea cadrului 0 ca mai înainte. În momentul în care se termină trimiterea a 26 de cadre, la $t = 520$, va sosi şi confirmarea pentru cadrul 0. Apoi, confirmările vor sosi la fiecare 20 milisecunde, aşa încât emiţătorul primeşte întotdeauna permisiunea să continue exact atunci când doreşte. În permanenţă există 25 sau 26 cadre neconfirmate. Cu alte cuvinte dimensiunea maximă a ferestrei emiţătorului este de 26.

Nevoia pentru o fereastră mare la emiţător apare atunci când produsul lărgime de bandă x timpul de propagare dus-întors este mare. Dacă lărgimea de bandă este mare, chiar şi pentru întârzieri moderate, emiţătorul îşi va termina repede fereastra. Daca întârzierea este mare (de exemplu, canal de satelit), emiţătorul îşi va termina fereastra chiar şi pentru lărgimi de bandă moderate. Produsul acestor doi factori spune de fapt care este capacitatea canalului, iar pentru a opera la eficienţă maximă, emiţătorul trebuie sa fie capabil sa o umple fără să se oprească.

Această tehnică este cunoscută ca **bandă de asamblare** (eng.: **pipelining**). Considerând capacitatea canalului de *b* biţi pe secundă, dimensiunea cadrului de *l* biţi şi timpul de propagare dus-întors *R* secunde, timpul necesar pentru a transmite un singur cadru este *l/b* secunde. După ce a fost transmis ultimul bit al unui cadru de date, apare o întârziere de *R/2* înainte ca biţii să ajungă la receptor şi o altă întârziere de cel puţin *R/2* pentru sosirea confirmării, rezultând o întârziere totală de *R*. În cazul protocoalelor pas-cu-pas, linia este ocupată pentru un timp egal cu *l/b* şi în aşteptare pentru un timp egal cu *R*, rezultând:

$$\text{utilizarea liniei} = l/(l+bR).$$

Dacă $l < bR$, eficienţa va fi mai mică de 50%. Deoarece până la întoarcerea confirmării există întotdeauna o întârziere nenulă, în principiu poate fi folosită banda de asamblare, pentru a ţine linia ocupată tot acest interval, dar dacă intervalul este mic, complexitatea suplimentară face efortul inutil.

Utilizarea benzii de asamblare în cazul unui canal de comunicaţie nesigur ridică probleme serioase. Mai întâi să vedem ce se întâmplă dacă un cadru din mijlocul unui şir lung este modificat sau pierdut. Multe cadre succesive vor ajunge la receptor înainte ca emiţătorul să observe că ceva

este greşit. Atunci când un cadru modificat ajunge la receptor este evident că el trebuie eliminat, dar ce trebuie să facă receptorul cu toate cadrele corecte care urmează? Să reamintim că nivelul legătură de date receptor este obligat să livreze pachete către nivelul reţea în secvenţă. În fig. 3-16, se prezintă efectele utilizării benzii de asamblare asupra revenirii în caz de eroare. Acum le vom examina în detaliu.

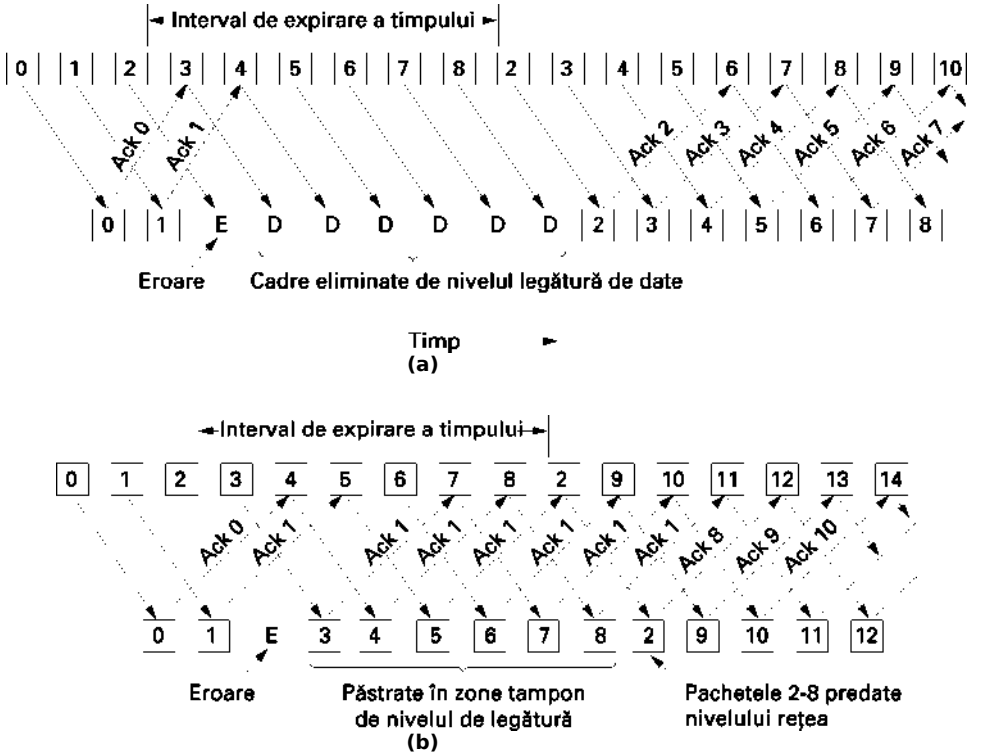


Fig. 3-16. Folosirea benzii de asamblare şi revenirea din eroare. Efectul unei erori când (a) dimensiunea ferestrei receptoare este 1 şi (b) dimensiunea ferestrei receptorului este mare.

Există două moduri de bază de tratare a erorilor în prezenţa benzii de asamblare. Un mod, numit **revenire cu n paşi** (eng.: **go back n**), este ca receptorul să elimine pur şi simplu cadrele care urmează, netrimiţând confirmări pentru cadrele eliminate. Această strategie corespunde unei ferestre de recepţie de dimensiune 1. Cu alte cuvinte, nivelul legătură de date refuză să accepte orice cadru exceptându-l pe următorul care trebuie livrat către nivelul reţea. Dacă fereastra emiţătorului se umple înaintea expirării contorului de timp, banda de asamblare va începe să se golească. În cele din urmă, timpul emiţătorului va expira şi se vor retransmite toate cadrele neconfirmate, în ordine, începând cu cadrul pierdut sau modificat. Dacă rata erorilor este mare, această abordare poate risipi o mare parte din lărgimea de bandă.

În fig. 3-16 (a) este prezentat protocolul de revenire cu n paşi pentru cazul în care fereastra receptorului are dimensiune unu. Cadrele 0 şi 1 sunt primite şi confirmate corect. Cadrul 2, totuşi, este alterat sau pierdut. Emiţătorul, care nu ştie de această problemă, continuă să trimită cadre până când timpul pentru cadrul 2 expiră. Apoi se întoarce la cadrul 2 şi o ia de la început cu el, trimiţând din nou cadrele 2, 3, 4 etc.

Cealaltă strategie generală de tratare a erorilor atunci când este folosită banda de asamblare se numeşte **repetare selectivă** (eng.: **selective repeat**). Când aceasta este utilizată, un cadru incorect este respins, dar toate cadrele corecte care îl urmează sunt memorate. Când contorul de timp al emiţătorului expiră, cel mai vechi cadru neconfirmat este retransmis. Dacă acest cadru ajunge corect, receptorul poate transmite către nivelul reţea, în ordine, cadrele pe care le-a memorat. Repetarea selectivă este deseori combinată cu utilizarea confirmărilor negative (NAK), care sunt trimise atunci când se detectează o eroare, de exemplu când se primeşte un cadru cu suma de control incorectă sau cu număr de secvenţă necorespunzător. Confirmările negative simulează retransmisia înainte de expirarea contorului de timp corespunzător, îmbunătăţind astfel performanţa.

În fig. 3-16 (b), cadrele 0 şi 1 sunt recepţionate corect, dar confirmarea pentru cadrul 2 este pierdută. Când cadrul 3 soseşte la receptor, nivelul legătură de date observă că a pierdut un cadru, şi trimite o confirmare negativă pentru 2, memorând însă cadrul primit. Când cadrele 4, 5 ajung la receptor, sunt la rândul lor memorate de nivelul legătură de date, în loc de a fi transmise nivelului reţea. În cele din urmă, confirmarea negativă pentru 2 ajunge înapoi la emiţător, care va retransmite cadrul 2. Când acesta ajunge la receptor, nivelul legătură de date are cadrele 2, 3, 4, 5, şi le poate pasa nivelului reţea în ordinea corectă. De asemenea, poate confirma toate cadrele până la 5 inclusiv, aşa cum se prezintă în figură. Dacă NAK-ul se pierde, în cele din urmă contorul de timp al emiţătorului va expira şi acesta va iniţia retransmisia, dar în acest fel se pierde mai mult timp. În concluzie, utilizarea confirmărilor negative accelerează retransmiterea unui anumit cadru.

Strategia de repetare selectivă corespunde unei ferestre a receptorului mai mare ca 1. Orice cadru din interiorul ferestrei poate fi acceptat şi memorat până când toate cele precedente vor fi trimise nivelului reţea. Dacă fereastra este mare, această abordare poate necesita un spaţiu mare de memorie pentru nivelul legătură de date.

Aceste două alternative reprezintă compromisuri între lărgimea de bandă şi spaţiul ocupat de tampoane la nivelul legătură de date. În funcţie de care resursă este mai deficitară, poate fi utilizată una sau cealaltă. Fig. 3-17 prezintă un protocol de tip bandă de asamblare în care nivelul legătură de date receptor acceptă cadre ordonate; cadrele ce urmează după o eroare sunt eliminate. În acest protocol, pentru prima dată, am renunţat la presupunerea că nivelul reţea are o rezervă infinită de pachete care trebuie trimise. Atunci când nivelul reţea are un pachet pe care doreşte să-l trimită, poate produce un eveniment *network_layer_ready.* Totuşi, pentru a impune regula de control al fluxului, conform căreia nu pot exista decât cel mult *MAX_SEQ* cadre neconfirmate, nivelul legătură de date trebuie să poată să interzică nivelului reţea să îl perturbe cu mai multe. Această funcţie este realizată

de funcţiile de bibliotecă *enable_network_layer* şi *disable_network_layer*.

Observaţi că în orice moment pot exista cel mult *MAX_SEQ* cadre şi nu *MAX_SEQ+*1 cadre neconfirmate, chiar dacă există *MAX_SEQ+1* numere de secvenţă: 0, 1, 2, *...MAX_SEQ.* Pentru a vedea de ce este necesară această restricţie, să considerăm următorul scenariu cu *MAX_SEQ* = 7.

1. Emiţătorul trimite cadrele de la 0 la 7.
2. O confirmare ataşată pentru cadrul 7 ajunge la emiţător.
3. Emiţătorul trimite alte opt cadre, din nou cu numerele de secvenţă de la 0 la 7.
4. Acum ajunge o altă confirmare ataşată pentru cadrul 7.

```
/* Protocolul 5 (revenire cu n paşi) permite mai multe cadre în aşteptare. Emiţătorul
poate trimite până la MAX_SEQ cadre fără a aştepta confirmare. În plus, spre deosebire
de protocoalele precedente, acesta nu presupune că nivelul reţea ar avea tot timpul un
nou pachet. În schimb, nivelul reţea provoacă un eveniment network_layer_ready atunci
când are de trimis un pachet */

#define MAX_SEQ 7                                        /* trebuie să fie 2^n – 1
*/
typedef enum {frame_arrival, cksum_err, timeout, network_layer_ready} event_type;
#include "protocol.h"

static boolean between(seq_nr a, seq_nr b, seq_nr c)
{
 /* întoarce adevărat dacă a <= b < c în mod circular şi fals în caz contrar */
 if (((a <= b) && (b < c)) || ((c < a) && (a <= b)) || ((b < c) && (c < a)))
   return(true);
 else
   return(false);
}

static void send_data(seq_nr frame_nr, seq_nr frame_expected, packet buffer[])
{
 /* construieşte şi trimite un cadru de date */
 frame s;                                                /* variabilă temporară
*/

 s.info = buffer[frame_nr]; /* inserează pachetul în cadru */ s.seq = frame_nr; /*
inserează numărul de secvenţă în cadru */ s.ack = (frame_expected + MAX_SEQ) %
(MAX_SEQ + 1); /* ataşează confirmarea */ to_physical_layer(&s); /* transmite cadrul
*/ start_timer(frame_nr); /* porneşte ceasul */ }

void protocol5(void)
{
 seq_nr next_frame_to_send;           /* MAX_SEQ > 1; utilizat pentru fluxul de ieşire
*/
 seq_nr ack_expected;                        /* cel mai vechi cadru încă neconfirmat
*/
 seq_nr frame_expected;               /* următorul cadru aşteptat, din fluxul de
intrare */
 frame r;                                                /* variabilă auxiliară
*/
 packet buffer[MAX_SEQ+1];                    /* zone tampon pentru fluxul de ieşire
*/
 seq_nr nbuffered;               /* număr de zone tampon de ieşire utilizate în
prezent */
```

```
    seq_nr i;                           /* utilizat ca index în vectorul de zone tampon
*/
    event_type event;
    enable_network_layer();                     /* permite evenimente network_layer_ready
*/
    ack_expected = 0;                               /* următoarea confirmare aşteptată
*/
    next_frame_to_send = 0;                             /* următorul cadru transmis
*/
    frame_expected = 0;                          /* numărul cadrului aşteptat să sosească
*/
    nbuffered = 0;                  /* iniţial în zonele tampon nu există nici un pachet
*/
    while(true) {
      wait_for_event(&event);           /* patru posibilităţi: vezi event_type, mai sus
*/
      switch(event) {
        case network_layer_ready:                /* nivelul reţea are un pachet de trimis
*/
          /*     acceptă,     salvează     şi     transmite     un     nou     cadru     */
          from_network_layer(&buffer[next_frame_to_send]);         /*  preia  noul  pachet
          */
          nbuffered = nbuffered + 1;                  /*  extinde  fereastra  emiţătorului
          */
          send_data(next_frame_to_send, frame_expected, buffer);   /* transmite cadrul */
          inc(next_frame_to_send);   /* creşte limita superioară a ferestrei emiţătorului
          */
          break;
        case frame_arrival:                     /* a sosit un cadru de date sau de control
*/
          from_physical_layer(&r);              /* preia de la nivelul fizic cadrul sosit
*/

          if (r.seq == frame_expected) {
            /* cadrele sunt acceptate doar în ordine */ to_network_layer(&r.info);      /*
                                                          predă pachetul nivelului reţea */
                    inc(frame_expected);/*   creste   limita   inferioară   a   ferestrei
          emiţătorului*/ }
          /* Confirmarea lui n implică n-l, n-2 etc. Verifică acest lucru. */
          while (between(ack_expected, r.ack, next_frame_to_send)) {
          /* tratează confirmarea ataşată */
          nbuffered = nbuffered - 1;              /* un cadru mai puţin în zonele tampon
*/
          stop_timer(ack_expected);             /* cadrul a sosit intact; opreşte ceasul
*/
          inc(ack_expected);                            /* contractă fereastra emiţătorului
*/
          }
          break;
        case cksum_err: break;            /* cadrele eronate sunt pur şi simplu ignorate
*/
        case timeout:                     /*    necaz;    retransmite    toate    cadrele
neconfirmate */
          next_frame_to_send = ack_expected;          /* începe retransmiterea de aici */
          for (i=1; i <= nbuffered; i++) {
```

```
            send_data(next_frame_to_send, frame_expected, buffer);/* retransmite 1
          cadru */ inc(next_frame_to_send);     /* pregăteşte transmiterea următorului
          */
        }

    }

    if (nbuffered < MAX_SEQ) enable_network_layer();
    else
      d i sable_network_layer();
 }
}
```

Fig. 3-17. Un protocol cu fereastră glisantă utilizând revenirea cu n paşi.

Întrebarea este: toate cele opt cadre aparţinând celui de al doilea lot au ajuns corect ori s-au pierdut în totalitate (considerând respingerile care urmează unei erori ca pierderi)? În ambele cazuri receptorul va trimite cadrul 7 ca o confirmare, deci emiţătorul nu are posibilitatea să ştie. Din acest motiv, numărul maxim de cadre neconfirmate trebuie limitat la *MAX_SEQ.*

Chiar dacă protocolul 5 nu păstrează cadrele sosite după o eroare, problema memorării nu dispare. Deoarece un emiţător poate avea de retransmis la un moment de timp viitor toate cadrele neconfirmate, el trebuie să păstreze toate cadrele transmise până când va fi sigur că au fost acceptate de receptor. Când soseşte o confirmare pentru cadrul *n,* cadrele *n*-l, *n*-2 ş.a.m.d. sunt confirmate automat. Această proprietate este foarte importantă atunci când unele dintre cadrele purtătoare de confirmări au fost pierdute sau modificate. De fiecare dată, când soseşte o confirmare, nivelul legătură de date verifică să vadă dacă unele tampoane pot fi eliberate. Dacă tampoanele pot fi eliberate (adică există spaţiu disponibil în fereastră), atunci nivelul reţea anterior blocat poate primi permisiunea să producă alte evenimente *network_layer_ready.*

În cazul acestui protocol, presupunem că există întotdeauna trafic în direcţia inversă de care să se poată ataşa confirmările. În caz contrar, confirmările nu pot fi trimise. Protocolul 4 nu are nevoie de această presupunere deoarece acesta trimite înapoi un cadru de fiecare dată când recepţionează unul. În protocolul următor, vom rezolva această problemă într-o manieră elegantă.

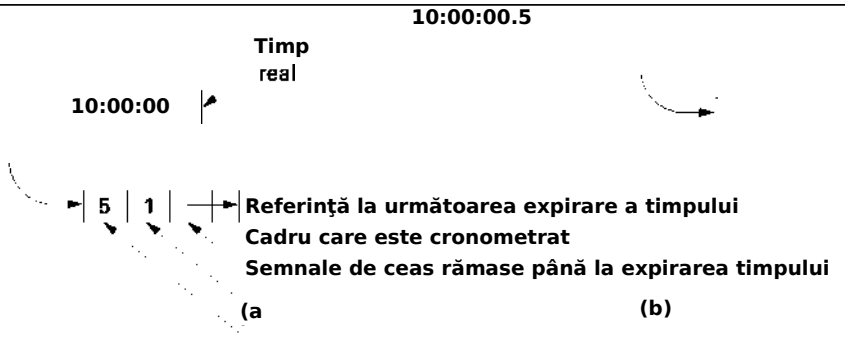


Fig. 3-18. Simularea prin program a contoarelor de timp multiple.

Deoarece protocolul 5 are mai multe cadre neconfirmate, este evident că necesită mai multe contoare de timp, câte unul pentru fiecare cadru neconfirmat. Timpul fiecărui cadru expiră independent de toate celelalte. Toate aceste contoare pot fi simulate uşor prin program, utilizând un singur ceas fizic care produce întreruperi periodice. Contoarele de timp active formează o listă înlănţuită, în fiecare nod existând informaţii despre câte semnale de ceas mai sunt până la expirarea timpului, cadrul care este cronometrat şi un pointer către următorul nod.

Ca o ilustrare a modului în care pot fi implementate contoarele de timp, să considerăm exemplul din fig. 3-18(a). Să presupunem că impulsurile de ceas sunt la fiecare 100 ms. Iniţial timpul real este 10:00:00.0; există trei timpi de expirare, la 10:00:00.5, 10:00:01.3 şi 10:00:01.9. La fiecare impuls de ceas produs de echipament, timpul real este actualizat şi contorul de impulsuri de la capătul listei este decrementat. Atunci când contorul de impulsuri de ceas devine zero, se produce o expirare de timp şi nodul este scos din listă, ca în fig. 3-18 (b). Chiar dacă această organizare cere ca lista să fie parcursă când este apelat *start_timer* sau *stop_ timer,* nu va necesita multe operaţii la fiecare impuls de ceas. În protocolul 5, ambele rutine au un parametru indicând pentru ce cadru se face contorizarea timpului.

3.4.3 Un protocol cu repetare selectivă

Protocolul 5 funcţionează bine dacă erorile sunt rare, dar dacă linia este slabă, se pierde mult din lărgimea de bandă cu retransmiterea cadrelor. O altă strategie de tratare a erorilor este ca receptorul să accepte şi să numeroteze cadrele care urmează după un cadru deteriorat sau pierdut. Un astfel de protocol nu elimină cadre doar pentru că un cadru anterior a fost deteriorat sau pierdut.

În acest protocol, atât emiţătorul cât şi receptorul menţin o fereastră de numere de secvenţă acceptabile. Dimensiunea ferestrei emiţătorului începe de la 0 şi creşte până la un maxim predefinit *MAX_SEQ.* Spre deosebire de aceasta, fereastra receptorului are dimensiunea fixă *MAX_SEQ.* Receptorul are un tampon rezervat pentru fiecare număr de secvenţă din cadrul ferestrei. Fiecare tampon are un bit asociat (*arrived – sosit*) care ne spune dacă tamponul este plin sau gol. De fiecare dată când soseşte un cadru, numărul său de secvenţă este verificat de funcţia

between, pentru a vedea dacă face parte din fereastră. Dacă da, şi dacă nu a fost deja recepţionat, este acceptat şi memorat. Această acţiune are loc fără să se verifice dacă conţine sau nu următorul pachet aşteptat de nivelul reţea. Desigur cadrul trebuie păstrat la nivelul legătură de date şi nu trebuie trimis către nivelul reţea decât atunci când toate cadrele cu numere mai mici au fost deja livrate nivelului reţea în ordinea corectă. Un protocol utilizând acest algoritm este prezentat în fig. 3 -19.

/* Protocolul 6 (repetare selectivă) acceptă cadrele în afara secvenţei, dar predă pachetele în ordine nivelului reţea. Fiecărui cadru neconfirmat îi este asociat un ceas. La expirarea timpului este retransmis doar acest cadru şi nu toate cele neconfirmate, ca în protocolul 5. */

```
#define MAX_SEQ 7                                        /* trebuie să fie 2"'n – 1 */
#define NR_BUFS ((MAX_SEQ + 1)/2)
typedef enum {frame_arrival, cksum_err, timeout, network_layer_ready, ack_timeout}
   event_type;
#include "protocol.h"
boolean no_nak = true;            /* încă nu a fost trimisă nici o confirmare negativă */
seq_nr oldest_frame = MAX_SEQ + 1;
static boolean between(seq_nr a, seq_nr b, seq_nr c)
{
/* La fel ca between din protocolul 5, dar mai scurt şi mai neclar. */
  return ((a <= b) && (b < c)) || ((c < a) && (a <= b)) || ((b < c) && (c < a));
}
static void send_frame(frame_kind fk, seq_nr frame_nr, seq_nr frame_expected, packet
buffer[])
{
  /* construieşte şi trimite un cadru de date, de ack sau de nak */
  frame s;                                                    /* variabilă temporară */
  s.kind = fk;                                             /* kind == data, ack sau nak */
  if (fk == data) s.info = buffer[frame_nr % NR_BUFS];
  s.seq = frame_nr;                              /* are sens doar pentru cadrele de date */
  s.ack = (frame_expected + MAX_SEQ) % (MAX_SEQ + 1);
  if (fk == nak) no_nak = false;                           /* un nak per cadru, te rog */
  to_physical_layer(&s);                                      /* transmite cadrul */
  if (fk == data) start_timer(frame_nr % NR_BUFS);
  stop_ack_timer();                         /* nu este nevoie de un cadru de ack separat */
}
void protocol6(void)
{
  seq_nr ack_expected; seq_nr next_frame_to_send; seq_nr frame_expected; seq_nr too_far;
  int   i;   frame   r;   packet   out_buf[NR_BUFS];   packet   în_buf[NR_BUFS];   boolean
  arrived[NR_BUFS];   seq_nr   nbuffered;   event_type   event;   enable_network_layer();
  ack_expected      =      0;        /* limita inferioară a ferestrei emiţătorului */ /*
  next_frame_to_send   =   0;        limita superioară a ferestrei emiţătorului + 1*/ /*
  frame_expected     =     0;           limita inferioară a ferestrei receptorului */ /*
  too_far     =     NR_BUFS;         limita superioară a ferestrei receptorului + 1*/ /*
  nbuffered = 0;                    indicele zonei tampon */ /* variabilă temporară */ /*
for (i = 0; i < NR_BUFS; i+         zone tampon pentru fluxul de ieşire */ /* zone tampon
+) while(true)                            pentru fluxul de intrare */ /* hartă de biţi de
{ wait_for_event();                 intrare */ /* câte zone tampon de ieşire sunt folosite
    switch(event) {                                                    în prezent */
      case
network_layer_ready:           /* iniţializează */ /* următoarea confirmare aşteptată în
        nbuffered           =  fluxul de intrare */ /* numărul următorului cadru transmis
nbuffered + 1;                                                          */
                                                          /* extinde fereastra */


                                        /* iniţial zonele tampon nu conţin nici un pachet
                                  */ arrived[i] = false;

                                        /* cinci variante: vezi event_type, mai sus
                                        */
```

```
        from_network_layer(&out_buf[next_frame_to_send % NR_BUFS]); /* preia un nou pachet
        */ send_frame(data, next_frame_to_send, frame_expected, out_buf); /* trimite
        cadrul */ inc(next_frame_to_send);     /* avansează limita superioară a ferestrei
        */
        break;
      case frame_arrival:                      /* a sosit un cadru de date sau de control
*/
        from_physical_layer(&r);               /* preia cadrul sosit de la nivelul fizic
*/
        if (r.kind == data) {
          /* A sosit un cadru nedeteriorat */
          if ((r.seq != frame_expected) && no_nak) send_frame(nak, 0, frame_expected,
            out_buf);
      else start_ack_timer();
          if (between(frame_expected, r.seq, too_far) && (arrived[r.seq%NR_BUFS]==false))
            { /* Cadrele pot fi acceptate în orice ordine */
            arrived[r.seq%NR_BUFS] = true;            /* marchează tamponul ca fiind plin
            */
            în_buf[r.seq%NR_BUFS] = r.info;                /* introduce datele în tampon
            */
            while (arrived[frame_expected % NR_BUFS]) {
            /* Predă cadrele şi avansează fereastra */
            to_network_layer(&în_buf[frame_expected % NR_BUFS]); no_nak = true;
            arrived[frame_expected % NR_BUFS] = false; inc(frame_expected);   /* avansează
            limita inferioară a ferestrei receptorului */
            inc(too_far);         /* avansează limita superioară a ferestrei receptorului
            */
            start_ack_timer();             /* pentru a stabili dacă e necesar ack separat
            */
            }
          }
        }
        if ((r.kind==nak)&&between(ack_expected, (r.ack+1)%
          (MAX_SEQ+1),next_frame_to_send)) send_frame(data, (r.ack+1)%(MAX_SEQ+1),
          frame_expected, out_buf);
        while (between(ack_expected, r.ack, next_frame_to_send)) {
          nbuffered = nbuffered − 1;                                  /* tratează ack ataşat
*/
          stop_timer(ack_expected % NR_BUFS);                  /* cadrul a ajuns intact
*/
          inc(ack_expected);      /*   avansează   marginea   inferioară   a   ferestrei
emiţătorului */
        } break;
      case cksum_err:
        if (no_nak) send_frame(nak, 0, frame_expected, out_buf);    /* cadru deteriorat */
        break;
      case timeout:
        send_frame(data, oldest_frame, frame_expected, out_buf);    /* a expirat timpul */
        break;
      case ack_timeout:
        send_frame(ack, 0, frame_expected, out_buf);                     /* timpul asociat
                                             confirmării pozitive a expirat; trimite ack
      */ }
      if (nbuffered ( NR_BUFS) enable_network_layer();
     else disable_network_layer();
```

```
    }
}
```

Fig. 3-19. Un protocol cu fereastră glisantă utilizând repetarea selectivă.

Recepţia nesecvenţială introduce anumite probleme ce nu sunt prezente în protocoalele în care cadrele sunt recepţionate numai în ordine. Putem ilustra problemele foarte uşor cu un exemplu. Să presupunem că avem un număr de secvenţă pe trei biţi şi deci emiţătorul poate transmite până la şapte cadre înainte să fie necesar să aştepte o confirmare. Iniţial ferestrele emiţătorului şi receptorului arată ca în fig. 3-20(a). Emiţătorul trimite acum cadrele de la 0 la 6. Fereastra receptorului îi permite să accepte orice cadru cu număr de secvenţă între 0 şi 6 inclusiv. Toate cele şapte cadre sosesc corect, deci receptorul le confirmă avansându-şi fereastra pentru a permite recepţia cadrelor 7, 0, 1, 2, 3, 4 sau 5, aşa cum arată fig. 3-20 (b). Toate cele 7 tampoane sunt marcate ca fiind goale.

În acest punct se produce dezastrul, din cauza unui fulger care loveşte linia telefonică, înlăturând toate confirmările. Emiţătorul ajunge în cele din urmă la timeout şi retransmite cadrul 0. Atunci când acest cadru soseşte la receptor, este făcută o verificare pentru a vedea dacă se încadrează în fereastra receptorului. Din păcate, în fig. 3-20(b) cadrul 0 este în interiorul noii ferestre şi deci va fi acceptat. Receptorul trimite o confirmare ataşată pentru cadrul 6, deoarece au fost recepţionate cadrele de la 0 la 6.

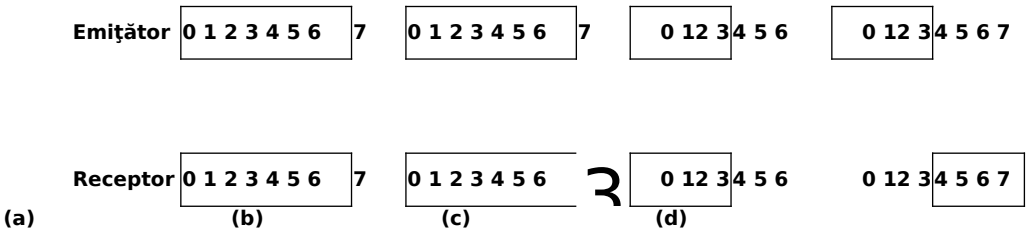


Fig. 3-20. (a) Situaţia iniţială cu o fereastră de dimensiune 7. (b) După ce şapte cadre au fost trimise şi recepţionate, dar neconfirmate. (c) Situaţia iniţială cu o fereastră de dimensiune patru. (d) După ce patru cadre au fost trimise şi recepţionate, dar neconfirmate.

Emiţătorul este fericit să afle că toate cadrele transmise au ajuns corect, deci îşi avansează fereastra şi trimite apoi imediat cadrele 7, 0, 1, 2, 3, 4 şi 5. Cadrul 7 va fi acceptat de receptor şi pachetul său va fi trimis direct nivelului reţea. Imediat după aceea, nivelul legătură de date receptor verifică să vadă dacă are un cadru 0 corect, descoperă că îl are şi trimite pachetul conţinut în el nivelului reţea. În consecinţă, nivelul reţea primeşte un pachet incorect şi protocolul eşuează.

Esenţa problemei este aceea că după ce receptorul îşi avansează fereastra, noul interval de numere de secvenţă corecte se suprapune peste cel vechi. Ca urmare, următoarea serie de cadre pot să fie sau duplicate (dacă toate confirmările sunt pierdute) sau unele noi (dacă toate confirmările au fost recepţionate). Receptorul nu are nici o posibilitate de a distinge cele două cazuri.

Pentru a ieşi din această dilemă, trebuie să ne asigurăm că atunci când

receptorul îşi deplasează fereastra, nu există nici o suprapunere peste cea anterioară. Pentru a asigura că nu există suprapunere, dimensiunea maximă a ferestrei trebuie să fie de cel mult jumătate din intervalul numerelor de secvenţă, aşa cum se face în fig. 3-20 (c) şi fig. 3-20 (d). De exemplu, dacă pentru numerele de secvenţă sunt utilizaţi 4 biţi, acestea vor lua valori de la 0 la 15. În orice moment trebuie să existe numai opt cadre. Astfel, dacă receptorul tocmai a acceptat cadrele de la 0 la 7 şi avansează fereastra pentru a permite acceptarea cadrelor de la 8 la 15, poate spune cu certitudine dacă următoarele cadre sunt retransmisii (de la 0 la 7) sau sunt unele noi (de la 8 la 15). În general, dimensiunea ferestrei pentru protocolul 6 va fi *(MAX_SEQ*+1)/2. Deci, pentru numere de secvenţă pe 4 biţi, dimensiunea ferestrei este 4.

O întrebare interesantă este: câte tampoane trebuie să aibă receptorul? În nici un caz el nu va accepta cadre ale căror numere de secvenţă sunt sub limita minimă a ferestrei, sau cadre ale căror numere de secvenţă depăşesc limita maximă a acesteia. În consecinţă, numărul de tampoane necesare este egal cu dimensiunea ferestrei, nu cu intervalul de valori al numerelor de secvenţă. În exemplul anterior, cu numere de secvenţă pe 4 biţi, sunt necesare 8 tampoane, numerotate de la 0 la 7. Atunci când soseşte cadrul *i,* acesta este pus în tamponul *i* mod 8. De notat că, deşi *i* şi (*i*+8) mod 8 „concurează" pentru acelaşi tampon, nu vor fi în aceeaşi fereastră simultan, deoarece aceasta ar implica o dimensiune a ferestrei de cel puţin 9.

Pentru acelaşi motiv, numărul de contoare de timp necesare este egal cu numărul de tampoane, nu cu dimensiunea spaţiului secvenţelor. Efectiv, există un contor de timp asociat fiecărui tampon. Atunci când contorul expiră, conţinutul tamponului este retransmis.

În protocolul 5, s-a presupus în mod implicit că acel canal este puternic încărcat. Când soseşte un cadru, nu se trimite imediat o confirmare. Confirmarea este ataşată la următorul cadru de date de ieşire. Dacă traficul invers este slab, confirmarea va fi reţinută o perioadă mare de timp. Dacă traficul este intens într-o direcţie şi inexistent în cealaltă direcţie, atunci sunt trimise numai *MAX_SEQ* cadre şi apoi protocolul se blochează, de aceea am presupus că exista întotdeauna trafic în direcţia inversă.

Această problemă este rezolvată în protocolul 6. După sosirea unei secvenţe de cadre cu date, este pornit un contor de timp auxiliar, prin *start_ack_timer.* Dacă până la expirarea acestui contor nu a apărut trafic în sens invers, atunci este trimis un cadru de confirmare separat. O întrerupere datorată contorului auxiliar se numeşte eveniment *ack_timeout.* Cu acest artificiu, fluxul de trafic unidirecţional este acum posibil, deoarece absenţa cadrelor de date în sens invers, pe care pot fi ataşate confirmări, nu mai este un obstacol. Există numai un contor auxiliar şi dacă *start_ack_timer* este apelată în timpul funcţionării contorului, acesta este resetat la un interval complet de timp de confirmare.

Este esenţial ca timpul de expirare asociat contorului auxiliar să fie mult mai scurt decât cel utilizat pentru cadrele de date de ieşire. Această condiţie este impusă pentru a ne asigura că o confirmare pentru un cadru corect recepţionat

soseşte înainte ca timpul emiţătorului să expire şi acesta să retransmită cadrul.

Protocolul 6 utilizează pentru tratarea erorilor o strategie mai eficientă decât protocolul 5. De fiecare dată când receptorul are motiv să suspecteze că a apărut o eroare, trimite înapoi la emiţător un cadru cu o confirmare negativă (NAK). Un asemenea cadru reprezintă o cerere pentru retransmiterea cadrului specificat în NAK. Există două cazuri în care receptorul ar trebui să fie suspicios: a sosit un cadru modificat sau a sosit un alt cadru decât cel aşteptat (un posibil cadru pierdut). Pentru a preveni producerea cererilor multiple de retransmisie a aceluiaşi cadru pierdut, receptorul va ţine minte dacă un NAK a fost deja trimis pentru un anumit cadru. Variabila *no_nak* din protocolul 6 are valoarea adevărat dacă nici un NAK nu a fost trimis pentru *frame_expected.* Dacă NAK a fost modificat sau pierdut, nu se întâmplă nimic, deoarece emiţătorul va ajunge, până la urmă, la timeout şi va retransmite cadrul lipsă. Dacă un cadru greşit soseşte după ce un NAK a fost transmis şi pierdut, *no_nak* va fi adevărat şi va fi pornit contorul de timp auxiliar. La expirarea acestuia, va fi trimis un ACK pentru resincronizarea emiţătorului cu starea curentă a receptorului.

În unele situaţii, timpul necesar pentru ca un cadru să se propage la destinaţie, să fie prelucrat şi să se recepţioneze confirmarea este (aproape) constant. În aceste condiţii, emiţătorul îşi poate ajusta contorul de timp să fie puţin mai mare decât intervalul de timp normal aşteptat între emiterea unui cadru şi recepţionarea confirmării sale. Totuşi, dacă acest timp variază puternic, emiţătorul trebuie să aleagă între fixarea intervalului la o valoare mică (riscând retransmisii inutile) şi fixarea la o valoare mare (rămânând în aşteptare timp îndelungat după producerea unei erori).

În ambele cazuri se iroseşte lărgime de bandă. Dacă traficul în sens invers este sporadic, timpul dinaintea confirmării va avea valori neregulate, fiind scurt când există trafic în sens invers şi lung când nu există. Variaţia timpului de prelucrare la receptor poate fi, de asemenea, o problemă. În general, atunci când deviaţia standard a intervalului de confirmare este mică, în comparaţie cu intervalul însuşi, intervalul de timp poate fi „strâmt” şi NAK-urile nu sunt utile. Altfel, contorul de timp trebuie să fie setat "larg" şi NAK-urile pot accelera apreciabil retransmisia cadrelor eronate sau pierdute.

Strâns legată de problema expirării timpului şi NAK-urilor este problema determinării cadrului care a cauzat expirarea timpului. În protocolul 5 acesta este întotdeauna *ack_expected,* deoarece este întotdeauna cel mai vechi. În protocolul 6 nu este uşor să se determine cel care a produs expirarea timpului. Să presupunem că au fost transmise cadrele de la 0 la 4, însemnând că lista cadrelor neconfirmate este 0, 1, 2, 3, 4, în ordinea de la cel mai vechi la cel mai nou. Acum să ne imaginăm că expiră timpul pentru 0, este transmis 5 (un nou cadru), expiră timpul pentru 1, expiră timpul pentru 2 şi este transmis 6 (un alt cadru nou). În acest moment, lista cadrelor neconfirmate este 3, 4, 0, 5, 1, 2, 6, de la cel mai vechi la cel mai nou. Dacă tot traficul de răspuns (mai precis, cadrele cu confirmări) este pierdut pentru un timp, expirarea timpului pentru cele şapte cadre neconfirmate se va produce în această ordine. Pentru a nu face ca exemplul să fie mai complicat decât este deja, nu am prezentat administrarea contoarelor de timp. În schimb, am presupus că, la

expirarea timpului, variabila *oldestjrame* este setată astfel, încât să indice cadrul pentru care a trecut timpul.

3.5 VERIFICAREA PROTOCOALELOR

Protocoalele reale şi programele ce le implementează sunt adesea destul de complicate. Ca urmare, a fost întreprinsă o imensă muncă de cercetare pentru a găsi tehnici formale, matematice, pentru specificarea şi verificarea protocoalelor. În secţiunile următoare vom studia câteva astfel de modele şi tehnici. Chiar dacă le privim în contextul nivelului legăturii de date, ele sunt, de asemenea, aplicabile şi altor niveluri.

3.5.1 Modele de tip automat finit

Un concept cheie folosit în multe modele de protocoale îl constituie **automatul finit**. Cu această tehnică, fiecare **automat al protocolului** (adică transmiţător sau receptor) este în fiecare moment de timp într-o stare specifică. Stările sale constau din toate valorile variabilelor sale, incluzând contorul de instrucţiuni al programului.

În cele mai multe cazuri, un număr mare de stări pot fi grupate împreună, în vederea analizei. De exemplu, considerând receptorul din protocolul 3, am putea abstractiza toate stările posibile în două stări importante: aşteptarea cadrului 0 sau aşteptarea cadrului 1. Toate celelalte stări pot fi considerate ca fiind tranzitorii, simpli paşi pe calea spre una din stările principale. De obicei, stările sunt alese ca fiind acele momente în care automatul protocolului aşteaptă să se petreacă următorul eveniment [adică să execute apelul de procedură *wait(event)* din exemplele noastre]. În acest punct, starea automatului este complet determinată de stările variabilelor sale. Numărul de stări este deci 2^n, unde n este numărul de biţi necesari pentru reprezentarea tuturor combinaţiilor de variabile.

Starea întregului sistem este combinaţia tuturor stărilor celor două automate ale protocolului şi a stării canalului. Starea canalului este determinată de conţinutul său. Folosind din nou protocolul 3 ca exemplu, canalul are patru stări posibile: un cadru zero sau un cadru unu circulând de la transmi- ţător la receptor, un cadru de confirmare circulând în sens invers sau nici un cadru. Dacă modelăm transmiţătorul sau receptorul prin două stări, întregul sistem are 16 stări distincte.

Aici trebuie să spunem câteva cuvinte despre starea canalului. Conceptul de cadru circulând „prin canal” este, bineînţeles, o abstractizare. Adevăratul înţeles este acela că este posibil ca respectivul cadru să fi fost primit, dar nu şi prelucrat la destinaţie. Un cadru rămâne „pe canal” până când automatul execută *FromPhysicalLayer* şi îl prelucrează.

Din fiecare stare, există zero sau mai multe **tranziţii** posibile spre alte stări. Tranziţiile au loc atunci când se petrece un eveniment. Pentru un automat, o

tranziţie trebuie să se facă atunci când este trimis un cadru, când soseşte un cadru, când expiră un interval de timp, când apare o întrerupere etc. Pentru canal, evenimentele tipice sunt introducerea unui nou cadru pe canal de către automatul protocolului, livrarea cadrului unui automat sau pierderea unui cadru datorată unei rafale de zgomote. Dată fiind o descriere completă a automatelor protocolului şi a caracteristicilor canalului, este posibil să trasăm graful orientat care prezintă toate stările automatului ca noduri şi toate tranziţiile ca arce orientate.

O singură stare este desemnată ca **stare iniţială**. Această stare corespunde descrierii sistemului, atunci când el începe să funcţioneze, sau unui punct de pornire convenabil imediat următor. Unele stări, poate chiar toate stările, pot fi atinse din starea iniţială printr-o secvenţă de tranziţii. Folosind tehnicile binecunoscute din teoria grafurilor (de exemplu, calculul închiderii tranzitive a unui graf), este posibil să se determine care stări sunt accesibile şi care nu. Această tehnică este numită **analiza accesibilităţii** (Lin ş.a., 1987). Această analiză poate fi utilă în determinarea corectitudinii protocolului.

Formal, un model de tip automat finit al unui protocol poate fi privit ca un cvadruplu *(S, M, I, T)* unde:

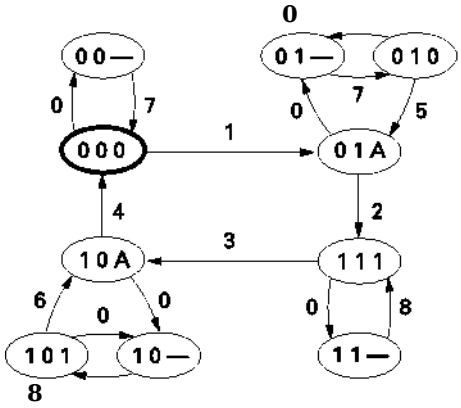
S este mulţimea stărilor în care se pot găsi procesele şi canalul
M este mulţimea cadrelor care pot fi schimbate prin canal
I este mulţimea stărilor iniţiale ale proceselor
T este mulţimea tranziţiilor între stări

La începutul intervalului de timp, toate procesele se găsesc în stările lor iniţiale. Apoi încep să se producă evenimente, cum ar fi disponibilizarea unor cadre pentru transmisie sau expirarea unor intervale de timp. Fiecare eveniment poate face ca unul dintre procese sau canalul să execute o acţiune şi să comute într-o nouă stare. Prin enumerarea atentă a fiecărui succesor posibil pentru fiecare stare, se poate construi graful de accesibilitate şi se poate analiza protocolul.

Analiza accesibilităţii poate fi folosită pentru a detecta diferite erori în specificaţia protocolului. De exemplu, dacă este posibil ca un anumit cadru să apară într-o anumită stare şi automatul finit să nu ştie ce acţiune trebuie întreprinsă, atunci specificaţia este eronată (incompletitudine). Dacă există o mulţime de stări fără ieşire şi din care nu se poate progresa, avem o altă eroare (interblocare). O eroare mai puţin serioasă este cea în care specificaţia protocolului spune cum să se trateze un eveniment într-o stare în care evenimentul nu se poate produce (tranziţie neesenţială). De asemenea pot fi detectate şi alte erori.

Ca exemplu de model de automat finit să considerăm fig. 3-21 (a). Acest graf corespunde protocolului 3 descris anterior: fiecare automat de protocol are două stări, iar canalul are patru stări. Există un total de 16 stări, nu toate accesibile din starea iniţială. Stările inaccesibile nu sunt reprezentate în figură. Fiecare stare este etichetată cu trei caractere, *SRC,* unde *S* este 0 sau 1, corespunzător cadrului pe care transmiţătorul (S) încearcă să îl expedieze; *R* este de asemenea 0 sau 1, corespunzător cadrului pe care receptorul (R) îl aşteaptă, iar *C* este 0, 1, *A* sau vid (-), corespunzător stării canalului. În acest exemplu, starea iniţială a fost aleasă ca fiind (000). Cu alte cuvinte, transmiţătorul tocmai a trimis cadrul 0, receptorul

aşteaptă cadrul 0 şi cadrul 0 este actualmente pe canal.



(a)

Tranziție	Cine rulează?	Cadru acceptat	Cadru emis	Spre nivelul rețea
0	—	(cadru pierdut)		—
1	R	0	A	Da
2	S	A	1	—
3	R	1	A	Da
4	S	A	0	—
5	R	0	A	Nu
6	R	1	A	Nu
7	S	(timp expirat)	0	—
8	S	(timp expirat)	1	—

(b)

Fig. 3-21. (a) Diagrama de stare pentru protocolul 3. (b) Tranziţiile.

În fig. 3-21 sunt prezentate nouă tipuri de tranziţii. Tranziţia 0 corespunde pierderii conţinutului canalului. Tranziţia 1 corespunde livrării corecte a pachetului 0 la receptor, receptorul schimbându- şi starea pentru a aştepta cadrul 1 şi emiţând o confirmare. Tranziţia 1 include şi livrarea pachetului 0 de către receptor spre nivelul reţea. Celelalte tranziţii sunt listate în fig. 3-21(b). Sosirea unui cadru cu suma de control eronată nu a fost pusă în evidenţă, deoarece nu trebuie schimbată starea (în protocolul 3).

Pe parcursul operării normale, tranziţiile 1, 2, 3 şi 4 sunt repetate în ordine, la nesfârşit. În fiecare ciclu sunt livrate două pachete, aducând transmiţătorul înapoi în starea iniţială, în care se încearcă transmiterea unui nou cadru cu numărul de secvenţă 0. În cazul în care canalul pierde cadrul 0, el face o tranziţie din starea (000) în starea (00-). Până la urmă, transmiţătorului îi expiră intervalul de timp (tranziţia 7) şi sistemul revine în starea (000). Pierderea unei confirmări este mai complicată, necesitând două tranziţii, 7 şi 5, sau 8 şi 6, pentru a repara eroarea.

Una din proprietăţile pe care protocolul cu număr de secvenţă pe 1 bit trebuie să le aibă este aceea că, indiferent de secvenţa de evenimente ce are loc, receptorul nu trebuie să livreze niciodată două pachete impare fără un pachet par intermediar şi invers. Din graful din fig. 3-21 se vede că această cerinţă poate fi formulată mai riguros astfel: „nu trebuie să existe căi din starea iniţială care să conţină două apariţii ale tranziţiei 1 fără ca între ele să apară tranziţia 3 sau invers.” Din figură se poate vedea că protocolul este corect în raport cu această cerinţă.

O cerinţă similară este aceea că nu trebuie să existe căi pe care transmiţătorul să-şi schimbe starea de două ori (de exemplu din 0 în 1 şi înapoi în 0) în timp ce starea receptorului rămâne constantă. Dacă ar exista o astfel de cale, atunci, în secvenţa corespunzătoare de evenimente, două cadre ar fi iremediabil pierdute, fără ca receptorul să observe. Secvenţa de pachete livrată ar avea în ea o pierdere nedetectată a două pachete.

O altă proprietate importantă a unui protocol este absenţa interblocărilor. O **interblocare** (eng.: **deadlock**) este situaţia în care protocolul nu mai înregistrează

nici un progres la transmitere (adică livrare de pachete spre nivelul reţea), indiferent de secvenţa de evenimente produse. În termenii modelului de graf, o interblocare este caracterizată de existenţa unei submulţimi de stări care este accesibilă din starea iniţială şi care are două proprietăţi:

1. Nu există nici o tranziţie într-o stare din afara submulţimii de stări.
2. În submulţimea de stări, nu există tranziţii care să determine continuarea transmiterii.

Odată ajuns în situaţia de interblocare, protocolul rămâne aici pentru totdeauna. Din nou, este uşor de văzut din graf că protocolul 3 nu are interblocări.

3.5.2 Modele de tip reţea Petri

Automatul finit nu este singura tehnică de specificare formală a protocoalelor. În această secţiune vom descrie o altă tehnică, **Reţelele Petri** (Danthine, 1980). O reţea Petri are patru elemente de bază: locuri, tranziţii, arce şi jetoane. Un **loc** reprezintă o stare în care se poate găsi sistemul (sau o parte a sa). Fig. 3-22 prezintă o reţea Petri cu două locuri, *A* şi *B,* reprezentate prin cercuri. Sistemul se află în starea *A*, indicată prin **jeton** (punctul îngroşat) în locul A. O **tranziţie** este indicată printr-o bară orizontală sau verticală. Fiecare tranziţie are zero sau mai multe **arce de intrare**, venind dinspre locuri de intrare, şi zero sau mai multe **arce de ieşire**, mergând spre locuri de ieşire.

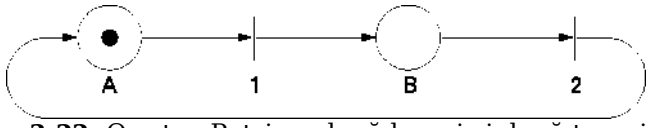


Fig. 3-22. O reţea Petri cu două locuri şi două tranziţii.

O tranziţie este **activată** dacă există cel puţin un jeton de intrare în fiecare din locurile sale de intrare. Orice tranziţie activată poate fi **executată** la dorinţă, ştergând un jeton din fiecare loc de intrare şi depunând un jeton în fiecare loc de ieşire. Dacă numărul de arce de intrare şi de ieşire diferă, jetoanele nu vor fi conservate. Dacă două sau mai multe tranziţii sunt activate, oricare din ele se poate executa. Alegerea tranziţiei care se va executa este nedeterministă, motiv pentru care reţelele Petri sunt utile în modelarea protocoalelor. Reţeaua Petri din fig. 3-22 este determinată şi poate fi folosită pentru a modela orice proces în două faze (de exemplu comportamentul unui bebeluş: mănâncă, doarme, mănâncă, doarme ş.a.m.d.). Ca în cazul tuturor instrumentelor de modelare, detaliile inutile sunt eliminate.

Fig. 3-23 dă modelul de tip reţea Petri pentru fig. 3-12. Spre deosebire de modelul de tip automat finit, aici nu există stări compuse: starea transmiţătorului, starea canalului şi starea receptorului sunt reprezentate separat. Tranziţiile 1 şi 2 corespund trimiterii cadrului 0 de către transmiţător, normal şi, respectiv, la expirarea timpului. Tranziţiile 3 şi 4 sunt analoagele pentru cadrul 1. Tranziţiile 5, 6 şi 7 corespund pierderii unui cadru 0, unei confirmări şi, respectiv, a unui cadru 1. Tranziţiile 8 şi 9 se petrec atunci când la receptor soseşte un cadru de date cu

număr de secvenţă greşit. Tranziţiile 10 şi 11 reprezintă sosirea la receptor a următorului cadrului din secvenţă şi livrarea acestuia către nivelul reţea.

Reţelele Petri pot fi reprezentate într-o formă algebrică convenabilă asemănătoare gramaticilor. Fiecărei tranziţii îi corespunde o regulă din gramatică. Fiecare regulă specifică locurile de intrare şi de ieşire ale tranziţiei. Din moment ce fig. 3-23 are 11 tranziţii, gramatica sa are 11 reguli, numerotate l-11, fiecare corespunzând tranziţiei cu acelaşi număr. Gramatica pentru reţeaua Petri din fig. 3-23 este următoarea:

1: BD AC
2: A A
3: AD BE
4: B B
5: C
6: D
7: E
8: CF DF
9: EG DG
10: CG DF
11: EF DG

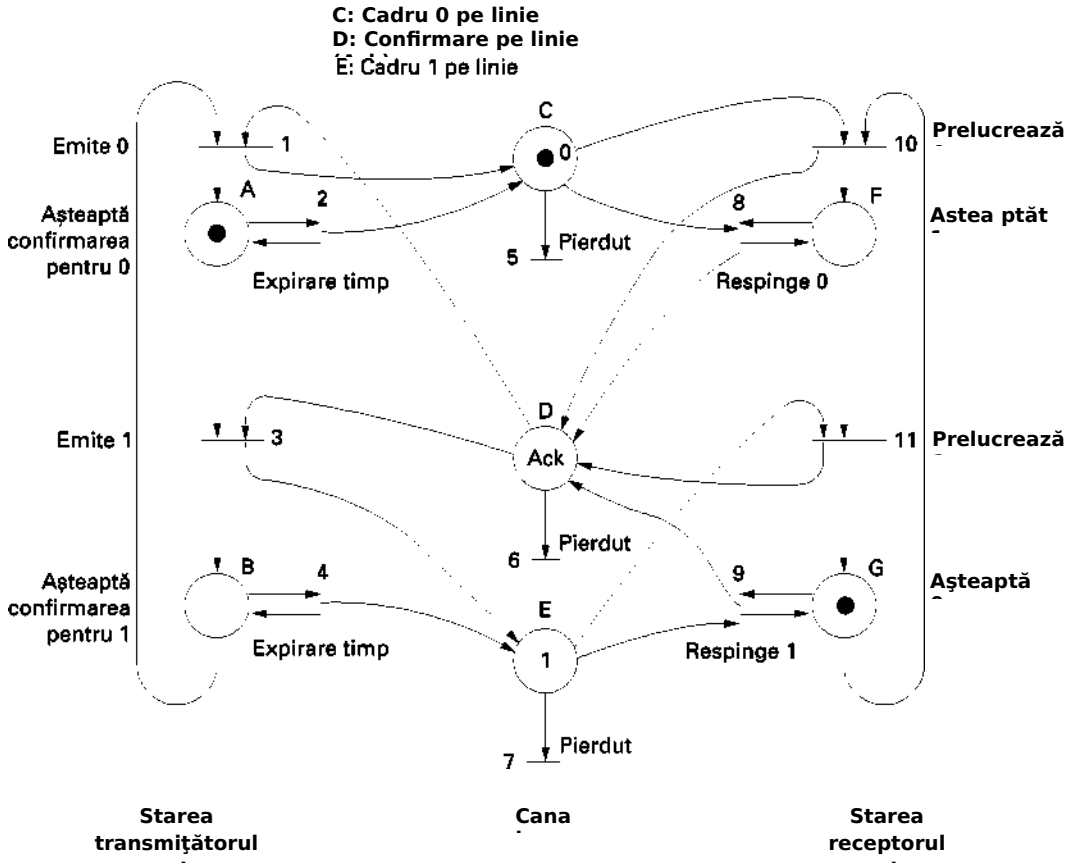


Fig. 3-23. Model de tip reţea Petri pentru

Reţelele Petri pot fi utilizate pentru a detecta erori în protocol, într-un mod similar folosirii automatelor finite. De exemplu, dacă o secvenţă executabilă a inclus tranziţia 10 de două ori fără a include tranziţia 11 ca intermediar, protocolul ar fi incorect. Conceputul de interblocare într-o reţea Petri este de asemenea similar corespondentului său de la automatul finit.

Interesant de observat este cum s-a reuşit reducerea unui protocol complex la 11 reguli gramaticale simple, care pot fi uşor manipulate de un program.

Starea curentă a reţelei Petri este reprezentată ca o colecţie neordonată de locuri, fiecare loc fiind reprezentat în colecţie de un număr de ori egal cu numărul de jetoane pe care le conţine. Orice regulă ale cărei locuri din membrul stâng sunt prezente, poate fi executată, ştergând aceste locuri din starea curentă şi adăugând locurile sale de ieşire la starea curentă. Marcajul din fig. 3-23 este ACG (adică A C şi G au fiecare câte un jeton), şi astfel, regula 10 *(CG^DF)* poate fi aplicată, ducând la o nouă stare (posibil cu acelaşi marcaj ca cea iniţială), dar regula 3 *(AD^-BE)* nu poate fi aplicată.

3.6 EXEMPLE DE PROTOCOALE ALE LEGĂTU RII DE DATE

În următoarele secţiuni vom examina câteva protocoale larg folosite pentru legătura de date. Primul dintre ele, HDLC, este un protocol clasic orientat pe bit ale cărui variante sunt folosite de decenii întregi în multe aplicaţii. Cel de-al doilea, PPP, este protocolul de nivel legătură de date folosite pentru conectarea calculatoarelor casnice la Internet.

3.6.1 HDLC – Controlul de nivel înalt al legăturii de date

În această secţiune vom examina un grup de protocoale strâns legate, puţin mai vechi, dar care sunt încă foarte utilizate. Ele sunt toate derivate din protocolul pentru legătura de date utilizat în lumea mainframe-urilor IBM, numit SDLC (**Synchronous Data Link Control**, rom.: protocolul de control sincron al legăturii de date). După ce a dezvoltat SDLC, IBM l-a supus examinării ANSI şi ISO pentru acceptare ca standard SUA şi, respectiv, internaţional. ANSI a modificat protocolul, astfel încât acesta a devenit **ADCCP** (**Advanced Data Communication Control Procedure**, rom.: procedură de control avansat al comunicaţiilor de date), iar ISO l-a modificat şi a produs **HDLC** (**High-level Data Link Control**, rom.: control de nivel înalt al legăturii de date). CCITT a adoptat şi modificat HDLC pentru al său **LAP** (**Link Access Procedure**, rom.: procedură de acces la legătură) care este parte a standardului pentru interfaţa de reţea X.25, dar, mai târziu l-a modificat din nou, rezultând **LAPB**, în scopul de a-l face mai compatibil cu o versiune ulterioară de HDLC. Un lucru frumos în ceea ce priveşte standardele este că sunt multe, dintre

care poţi alege. În plus, dacă nu îţi place nici unul dintre ele, poţi aştepta modelul care va apărea anul viitor.

Aceste protocoale se bazează pe aceleaşi principii. Toate sunt orientate pe biţi şi folosesc inserarea de biţi pentru transparenţa datelor. Ele diferă doar în puncte minore, şi totuşi supărătoare. Discuţia care urmează, despre protocoalele orientate pe biţi, intenţionează a fi o introducere generală. Pentru detaliile specifice fiecărui protocol, consultaţi definiţia corespunzătoare.

Toate protocoalele orientate pe biţi folosesc structura de cadru prezentată în fig. 3-24. Câmpul *Adresă* este primul ca importanţă pentru liniile cu terminale multiple, unde el este folosit pentru a identifica unul dintre terminale. Pentru liniile punct-la-punct, el este folosit uneori pentru a deosebi comenzile de răspunsuri.

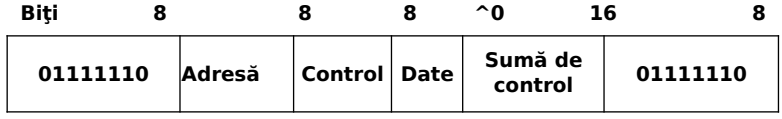


Fig. 3-24. Format de cadru pentru protocoalele orientate pe biţi.

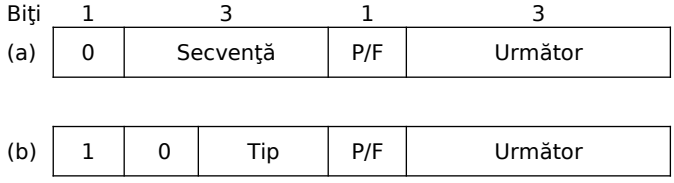
Câmpul *Control* este folosit pentru numere de secvenţă, confirmări şi alte scopuri, după cum se va arăta în continuare.

Câmpul *Date* poate conţine informaţii arbitrare. Poate avea lungime arbitrară, cu toate că eficienţa sumei de control scade odată cu creşterea lungimii cadrului, datorită creşterii probabilităţii de apariţie a erorilor în rafală.

Câmpul *Suma de Control* este o variantă CRC (Cyclic Redundancy Code – cod ciclic redundant), folosind tehnica prezentată în Sec. 3-2.2.

Cadrul este delimitat cu o altă secvenţă indicator (01111110). Pe liniile punct-la-punct inactive secvenţele indicator sunt transmise continuu. Un cadru minim conţine trei câmpuri şi are în total 32 de biţi, excluzând indicatorii de la capete.

Există trei tipuri de cadre: **Informaţie**, **Supervizor** şi **Nenumerotat**. Conţinutul câmpului *Control* pentru fiecare dintre aceste trei tipuri este prezentat în fig. 3-25. Acest protocol foloseşte o fereastră glisantă, cu un număr de secvenţă reprezentat pe 3 biţi. În fereastră pot fi păstrate, la un moment dat, până la şapte cadre neconfirmate. Câmpul *Secvenţă* din fig. 3-25(a) este numărul de secvenţă al cadrului. Câmpul *Următor* este o confirmare ataşată. Oricum, toate protocoalele aderă la convenţia că, în loc să ataşeze numărul ultimului cadru recepţionat corect, să folosească numărul primului cadru nerecepţionat (adică următorul cadru aşteptat). Opţiunea pentru ultimul cadru primit sau următorul cadru recepţionat este arbitrară; nu are importanţă ce convenţie este utilizată, dacă este folosită cu consecvenţă.



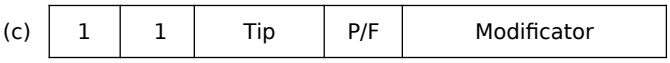


Fig. 3-25. Câmpul Control pentru (a) un cadru de informaţie, (b) un cadru de supervizare,
(c) un cadru nenumerotat.

Bitul *P/F* înseamnă *Test/Final (eng.: Poll/Final).* El este folosit atunci când un calculator (sau un concentrator) interoghează un grup de terminale. Când este folosit ca *P*, calculatorul invită terminalul să trimită date. Toate cadrele trimise de terminal, cu excepţia celui final, au bitul *P/F* setat pe **P**. Pentru cadrul final bitul este setat la *F*.

În câteva dintre protocoale, bitul *P/F* este folosit pentru a forţa cealaltă maşină să trimită imediat un cadru Supervizor, în loc să aştepte fluxul invers la care să se ataşeze informaţia despre fereastră. Bitul are de asemenea câteva utilizări minore referitoare la cadrele nenumerotate.

Numeroasele tipuri de cadre Supervizor sunt diferenţiate prin câmpul *Tip.* Tipul 0 este un cadru de confirmare (numit oficial RECEIVE READY) folosit pentru a indica următorul cadru aşteptat. Cadrul este folosit atunci când nu există flux invers care să poată fi folosit pentru ataşare.

Tipul 1 este un cadru de confirmare negativă (oficial numit REJECT). Este folosit pentru a indica detecţia unei erori de transmisie. Câmpul *Următor* indică primul cadru din secvenţă ce nu a fost recepţionat corect (deci cadrul ce trebuie retransmis). Transmiţătorului i se cere să retransmită toate cadrele neconfirmate, începând cu *Următor-ul.* Această strategie este similară mai degrabă protocolului 5 decât protocolului 6.

Tipul 2 este RECEIVE NOT READY. El confirmă toate cadrele, cu excepţia lui *Următor,* exact ca RECEIVE READY, dar spune transmiţătorului să oprească transmisia. RECEIVE NOT READY este destinat să semnaleze anumite probleme temporare apărute la receptor, cum ar fi lipsa zonelor tampon, şi nu ca o alternativă la controlul fluxului cu fereastră glisantă. Când problema a fost rezolvată, receptorul trimite un RECEIVE READY, REJECT sau anumite cadre de control.

Tipul 3 este SELECTIVE REJECT. El cere retransmiterea, însă doar pentru cadrul specificat. Din acest punct de vedere este mai apropiat de protocolul 6 decât de protocolul 5 şi de aceea este folositor atunci când dimensiunea ferestrei transmiţătorului este jumătate sau mai puţin din dimensiunea spaţiului secvenţei. Astfel, dacă receptorul doreşte să păstreze cadre care erau în afara secvenţei pentru posibila folosire ulterioară, el poate să forţeze retransmiterea oricărui cadru, folosind SELECTIVE REJECT. HDLC şi ADCCP permit acest' tip de cadru, dar SDLC şi LAPB nu îl permit (adică nu există Selective Reject) şi cadrele de tipul 3 nu sunt definite.

Cea de-a treia clasă o reprezintă cadrul Nenumerotat. El este folosit uneori în scopuri de control, dar poate fi folosit şi pentru transportul datelor atunci când se recurge la un serviciu nesigur, neorientat pe conexiune. Diversele tipuri de protocoale orientate pe biţi diferă considerabil aici, spre deosebire de celelalte două tipuri, unde erau aproape identice. Pentru a indica tipul cadrului sunt disponibili cinci biţi, dar nu sunt folosite toate cele 32 de posibilităţi.

Toate protocoalele furnizează o comandă, DISC (DISConnect), care permite ca o maşină să anunţe că se va opri (de exemplu pentru întreţinere preventivă). De asemenea există o comandă ce permite ca o maşină, care tocmai s-a reconectat, să-şi anunţe prezenţa şi să forţeze resetarea tuturor numerelor de secvenţă la zero. Această comandă poartă numele de SNRM (Set Normal Response Mode – stabileşte modul normal de răspuns). Din nefericire, „modul normal de răspuns" numai normal nu este. Este un mod neechilibrat (adică asimetric) în care unul din capetele liniei este master iar celălalt este slave. SNRM datează din timpurile când comunicaţia datelor presupunea un terminal neinteligent comunicând cu un calculator gazdă puternic, ceea ce este, evident, asimetric. Pentru a face protocolul mai potrivit cazurilor în care cei doi parteneri sunt egali, HDLC şi LAPB au o comandă suplimentară, SABM (Set Asynchronous Balanced Mode – stabileşte modul asincron echilibrat), care resetează linia şi declară ambii parteneri ca fiind egali. De asemenea, aceste protocoale au comenzile SABME şi SNRME, care sunt identice cu SABM şi, respectiv, SNRM, cu excepţia faptului că ele permit folosirea unui format extins pentru cadru, care utilizează numere de secvenţă pe 7 biţi în locul unora pe 3 biţi.

A treia comandă prevăzută de toate protocoalele este FRMR (FRaMe Reject), folosită pentru a indica sosirea unui cadru cu suma de control corectă, dar cu semantică imposibilă. Exemple de semantică imposibilă sunt cadru de tipul 3 Supervizor în LAPB, un cadru mai scurt de 32 de biţi, un cadru de control nepermis, confirmarea unui cadru care a fost în afara ferestrei etc. Cadrele FRMR conţin un câmp de date de 24 de biţi care arată ceea ce a fost eronat la cadrul respectiv. Datele includ câmpul de control al cadrului eronat, parametrii ferestrei şi o colecţie de biţi folosiţi pentru a semnala erori specifice.

Cadrele de control pot fi pierdute sau deteriorate ca şi cadrele de date, de aceea şi ele trebuie confirmate. În acest scop este furnizat un cadru special de control, numit UA (Unnumbered Acknowledgement). Deoarece poate exista un singur cadru de control neconfirmat, nu există niciodată ambiguităţi asupra cadrului care este confirmat.

Cadrele de control rămase sunt folosite pentru iniţializare, interogare şi raportarea stării. Există, de asemenea, un cadru de control care poate conţine informaţii arbitrare, UI (Unnumbered Information). Aceste date nu sunt livrate nivelului reţea, ci sunt destinate a fi primite chiar de nivelul legătură de date.

În ciuda utilizării pe scară largă, HDLC este departe de a fi perfect. O discuţie despre diversitatea problemelor asociate cu acest protocol poate fi găsită în (Fiorini ş.a., 1995).

3.6.2 Nivelul legăturii de date în Internet

Internet-ul constă din maşini individuale (calculatoare gazdă şi rutere) şi o infrastructură de comunicaţie care le conectează. În cadrul unei singure clădiri sunt larg utilizate LAN-urile pentru interconectare, dar infrastructura de arie largă este construită din linii închiriate, punct-la-punct. În Cap. 4 vom studia LAN-urile; aici vom examina protocoalele legăturii de date folosite pe liniile punct-la-punct în

Internet.

În practică, comunicaţia punct-la-punct este folosită în principal în două situaţii. În primul rând, mii de organizaţii au una sau mai multe LAN-uri, fiecare cu un anumit număr de calculatoare gazdă (calculatoare personale, staţii de lucru ale utilizatorilor, servere ş.a.m.d.) şi un ruter (sau o punte care este funcţional similară). Adeseori, ruterele sunt interconectate printr-un trunchi LAN. În mod tipic, toate conexiunile cu lumea exterioară se fac printr-unul sau două rutere care au linii punct-la- punct închiriate spre rutere aflate la distanţă. Internet-ul este construit din aceste rutere şi liniile lor închiriate care realizează subreţelele de

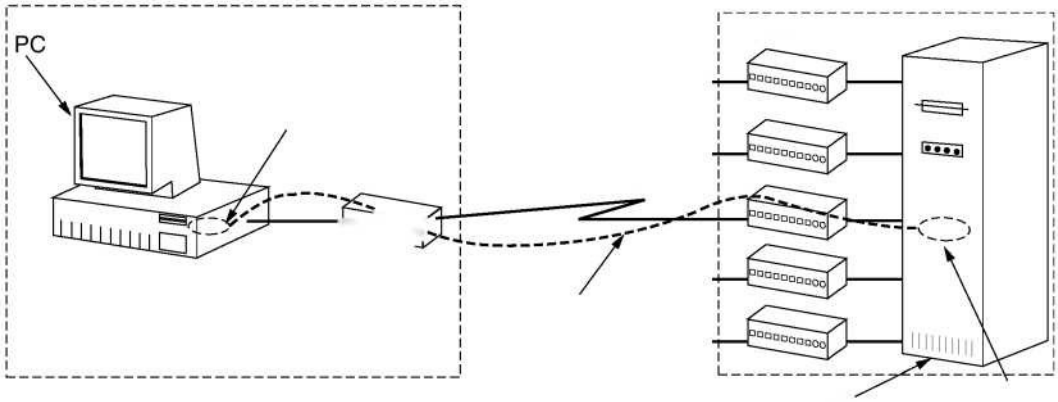


Fig. 3-26. Un calculator personal lucrând ca un calculator

comunicaţie.

A doua situaţie în care liniile punct-la-punct joacă un rol major în Internet o reprezintă milioanele de utilizatori individuali care au conexiuni de acasă la Internet folosind modemuri şi linii telefonice comutate. De obicei, PC-ul de acasă al unui utilizator apelează ruterul unui **furnizor de servicii Internet**, şi poate acţiona astfel ca orice calculator gazdă Internet. Această metodă de operare nu este diferită de aceea în care există o linie închiriată între PC şi ruter, cu excepţia faptului că atunci când utilizatorul îşi termină sesiunea, conexiunea va fi închisă. În fig. 3-26 este ilustrat un PC casnic ce apelează un furnizor de servicii Internet. Modem-ul este prezentat ca fiind extern, tocmai pentru a-i accentua rolul, dar calculatoarele moderne dispun de modem-uri interne.

Atât pentru conexiunea pe linie închiriată ruter-ruter cât şi pentru conexiunea comutată calculator gazdă-ruter, este folosit un protocol de legătură de date punct-la-punct pentru încadrare, controlul erorilor şi pentru alte funcţii ale nivelului legătură de date pe care le-am studiat în acest capitol. Cel folosit în Internet este PPP. În continuare acesta va fi prezentat în detaliu.

PPP – Point-to-Point Protocol (rom.: protocol punct-la-punct)

Internetul are nevoie de un protocol punct-la-punct care să servească mai multor scopuri, incluzând trafic ruter-la-ruter şi trafic utilizator-la-ISP. Acest protocol este

PPP (**Point-to-Point Protocol**, rom.: **protocolul punct-la-punct**), care este definit în RFC 1661 şi dezvoltat în alte câteva RFC-uri (de exemplu RFC-urile 1662, 1663). PPP face detecţia erorilor, suportă mai multe protocoale, permite ca adresele IP să fie negociate în momentul conectării, permite autentificarea şi are multe alte capabilităţi.

PPP furnizează trei lucruri:

1. O metodă de împărţire în cadre care delimitează, fără ambiguitate, sfârşitul unuia şi începutul următorului. Formatul cadrului permite şi detecţia de erori.
2. Un protocol de control al legăturii pentru a obţine liniile, a le testa, a negocia opţiunile şi pentru a elibera liniile atunci când nu mai este nevoie de ele. Acest protocol se numeşte **LCP**(**Link Control Protocol**, rom: **protocolul de control al legăturii**). El suportă circuite sincrone şi asincrone şi codificări orientate atât pe bit, cât şi pe caracter.
3. Un mod de a negocia opţiunile nivelului reţea într-un mod independent de protocolul folosit pentru nivelul reţea. Metoda aleasă este de a avea un **NCP** (**Network Control Protocol**, rom: **protocol de control al reţelei**) pentru fiecare nivel de reţea suportat.

Pentru a vedea cum lucrează împreună aceste părţi, să considerăm un scenariu tipic în care un utilizator sună de la domiciliu un furnizor de servicii Internet pentru a transforma PC-ul său de acasă într-un calculator gazdă Internet temporar. PC-ul apelează mai întâi ruterul furnizorului prin intermediul unui modem. După ce modemul ruterului a răspuns la telefon şi s-a stabilit o conexiune fizică, PC-ul trimite ruterului o serie de pachete LCP în câmpul de informaţie utilă (payload) al unuia sau mai multor cadre PPP. Aceste pachete şi răspunsurile lor selectează parametrii PPP ce vor fi utilizaţi.

Odată ce parametrii s-au stabilit de comun acord, mai multe pachete NCP sunt trimise pentru a configura nivelul reţea. În mod obişnuit, PC-ul vrea să ruleze o suită de protocoale TCP/IP şi va avea nevoie de o adresă IP. Deoarece nu există adrese IP suficiente, fiecare furnizor de Internet ia o parte din ele şi asociază dinamic câte una pentru fiecare PC ataşat în reţea, pe durata sesiunii de conectare. Dacă un furnizor posedă n adrese IP, el poate avea până la n maşini conectate simultan, dar numărul total de clienţi poate fi de mai multe ori pe atât. NCP pentru IP este folosit pentru a realiza asocierea adreselor IP.

În acest moment, PC-ul este un calculator gazdă Internet şi poate trimite şi primi pachete IP, exact aşa cum o pot face calculatoarele conectate prin cabluri. Când utilizatorul termină, NCP întrerupe conexiunea la nivelul reţea şi eliberează adresa IP. Apoi LCP întrerupe conexiunea la nivelul legătură de date. În final, calculatorul spune modemului să închidă telefonul, eliberând conexiunea la nivel fizic.

Formatul cadrului PPP a fost ales foarte asemănător cu formatul cadrului HDLC deoarece nu exista nici un motiv pentru a se reinventa roata. Diferenţa majoră între PPP şi HDLC este că primul este mai degrab ă orientat pe caractere decât pe biţi. În particular, PPP foloseşte umplerea cu caractere pe liniile comutate prin modem, astfel încât toate cadrele au un număr întreg de octeţi. Nu este posibil să se trimită un cadru constând din 30.25 octeţi, aşa cum era la HDLC. Cadrele PPP pot fi

transmise nu numai pe liniile telefonice comutate, ele pot fi transmise şi pe linii SONET sau linii HDLC, cu adevărat orientate pe biţi (de exemplu pentru conexiuni ruter-ruter). Formatul cadrului PPP este prezentat în fig. 3-27.

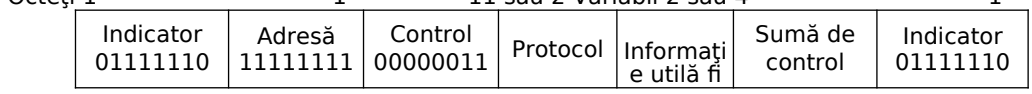


Fig. 3-27. Formatul complet de cadru PPP pentru operarea în mod nenumerotat.

Toate cadrele PPP încep cu octetul indicator HDLC standard (01111110), pentru care se foloseşte umplerea cu caractere, dacă apare în cadrul câmpului ce specifică informaţia utilă. După acesta urmează câmpul *Adresă,* care este întotdeauna setat la valoarea binară 11111111, indicând astfel că toate staţiile trebuie să accepte cadrul. Folosirea acestei valori evită problema necesităţii de a se asocia adrese legăturii de date.

Câmpul *Adresă* este urmat de câmpul *Control,* a cărui valoare implicită este 00000011. Această valoare indică un cadru nenumerotat. Cu alte cuvinte, PPP nu oferă o transmisie sigură folosind numere de secvenţă şi confirmări în mod implicit. În medii cu zgomote, cum ar fi reţelele fără fir, poate fi folosită transmisia sigură utilizând numere de secvenţă. Detaliile exacte sunt definite în RFC 1663, dar această facilitate este rar utilizată.

Deoarece câmpurile *Adresă* şi *Control* sunt întotdeauna constante în configuraţiile implicite, LCP furnizează mecanismul necesar ca cele două părţi să negocieze opţional omiterea amândurora şi să economisească astfel doi octeţi pe cadru.

Cel de-al patrulea câmp PPP este câmpul *Protocol.* Sarcina lui este să spună ce tip de pachet este în câmpul *Informaţie utilă.* Sunt definite coduri pentru LCP, NCP, IP, IPX, AppleTalk şi alte protocoale. Protocoalele ce încep cu un bit 0 sunt protocoale pentru nivelul reţea. cum ar fi IP, IPX, OSI CLNP, XNS. Acelea care încep cu un bit 1 sunt folosite pentru a negocia alte protocoale. Acestea includ LCP şi un NCP diferit pentru fiecare protocol de reţea suportat. Dimensiunea implicită a câmpului *Protocol* este de 2 octeţi, dar ea poate fi negociată la 1 octet folosind LCP.

Câmpul *Informaţie utilă* este de lungime variabilă, până la o anumită limită maximă negociată. Dacă lungimea nu este negociată folosind LCP în timpul setării liniei, este folosită o lungime implicită de 1500 de octeţi. Dacă este necesar, după informaţia utilă pot fi adăugate caractere de umplere.

După câmpul *Informaţie utilă* urmează câmpul S*umă de control*, care este în mod normal de 2 octeţi, dar poate fi modificat la 4 octeţi.

În concluzie, PPP este un mecanism de încadrare multiprotocol potrivit pentru folosirea pe linii cu modem, linii seriale orientate pe biţi HDLC, SONET şi alte niveluri fizice. Suportă detecţia erorilor, negociere opţională, compresia antetului şi, opţional, transmisie sigură folosind cadre HDLC.

Să ne întoarcem acum de la formatul cadrului PPP la modul în care liniile sunt stabilite (eng.: brought up) şi eliberate (eng.: brought down). Diagrama simplificată din fig. 3-28 arată fazele prin care trece o linie atunci când este stabilită, folosită şi eliberată. Secvenţa se aplică atât pentru conexiunile prin modem cât şi pentru conexiunile ruter-ruter.

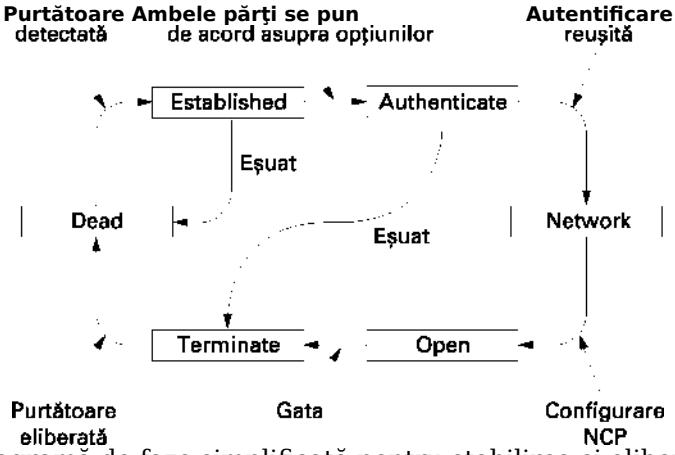


Fig. 3-28. O diagramă de faze simplificată pentru stabilirea şi eliberarea unei linii.

Protocolul se iniţializează cu linia în starea *DEAD,* stare care semnifică faptul că nu este prezentă nici o purtătoare la nivel fizic şi nu există nici o conexiune fizică. După ce este stabilită conexiunea fizică, linia trece în *ESTABLISH.* În acest punct începe negocierea opţională LCP care, dacă reuşeşte, conduce la *AUTHENTICATE.* Acum cele două părţi pot să-şi verifice una alteia identitatea, dacă doresc. Când se intră în faza *NETWORK,* este invocat protocolul NCP corespunzător pentru a configura nivelul reţea. Dacă configurarea se face cu succes, este atinsă faza *OPEN* şi poate avea loc transportul datelor. Când transportul datelor este terminat, linia este trecută în faza *TERMINATE* şi, de aici, înapoi în *DEAD* unde purtătoarea este întreruptă.

Nume	Direcţie	Descriere
Configure-request	I^R	Lista opţiunilor şi valorilor propuse
Configure-ack	I^R	Toate opţiunile sunt acceptate
Configure-nak	I^R	Anumite opţiuni nu sunt acceptate
Configure-reject	I^R	Anumite opţiuni nu sunt negociabile
Terminate-request	I^R	Cerere de eliberare a liniei
Terminate-ack	I^R	OK, linia este eliberată
Code-reject	I^R	Primire cerere necunoscută
Protocol-reject	I^R	Cerere protocol necunoscut
Echo-request	I^R	Rog trimiterea acestui cadru înapoi

Echo-replay	I^R	Iată cadrul înapoi
Discard-request	I^R	Ignoră cadrul (pentru testare)

Fig. 3-29. Tipurile de cadre LCP.

LCP negociază opţiunile protocolului legăturii de date în timpul fazei *ESTABLISH.* Protocolul LCP nu se ocupă chiar de opţiuni, ci de mecanismul de negociere. El furnizează procesului iniţiator un mod de a face o propunere şi procesului de răspuns un mod de a accepta sau refuza această propunere. De asemenea, el furnizează celor două procese un mecanism de a testa calitatea liniei, de a verifica dacă aceasta este suficient de bună pentru a defini o conexiune. În fine, protocolul LCP permite liniilor să fie eliberate atunci când nu mai este nevoie de ele.

În RFC 1661 sunt definite unsprezece tipuri de cadre LCP. Acestea sunt listate în fig. 3-29. Cele patru tipuri *Configure-* permit iniţiatorului (I) să propună valori pentru opţiuni şi celui care răspunde (R) să le accepte sau să le refuze. În ultimul caz, cel care răspunde poate face o propunere alternativă sau poate anunţa că nu este gata să negocieze în nici un fel anumite opţiuni. Opţiunile ce vor fi negociate şi valorile propuse pentru ele sunt conţinute în cadrele LCP.

Codurile *Terminate-* sunt folosite pentru a elibera o linie atunci când ea nu mai este necesară. Codurile *Code-reject* şi *Protocol-reject* sunt folosite de către cel ce răspunde pentru a spune că a primit ceva ce nu înţelege. Această situaţie poate însemna că a avut loc o eroare de transmisie, dar, mai degrabă, înseamnă că iniţiatorul şi cel ce răspunde folosesc versiuni diferite ale protocolului LCP. Tipurile *Echo-* sunt folosite pentru a testa calitatea liniei. În sfârşit, *Discard-request* este folosit pentru depanare. Dacă unul din capete are probleme cu transmiterea biţilor, programatorul poate folosi acest tip pentru testare. Dacă el reuşeşte să meargă de la un capăt la celălalt, receptorul doar îl rejectează, fără a întreprinde nici o acţiune care ar putea genera confuzii pentru persoana care testează.

Opţiunile care pot fi negociate includ definirea dimensiunii maxime pentru informaţia utilă din cadrele de date, activarea autentificării şi alegerea protocolului ce va fi folosit, activarea monitorizării calităţii liniei în timpul operaţiunilor normale şi selectarea diferitelor opţiuni pentru comprimarea antetului.

Nu se pot spune multe despre protocoalele NCP în general. Fiecare este specific unui anumit protocol de nivel reţea şi permite să se facă cereri de configurare ce sunt specifice unui anumit protocol. De exemplu, pentru IP, asocierea dinamică a adreselor este cea mai importantă posibilitate.

3.7 REZUMAT

Sarcina nivelului legătură de date este de a converti şirurile de biţi oferite de nivelul fizic în şiruri de cadre pentru a fi folosite de către nivelul reţea. Sunt utilizate diferite metode de încadrare, incluzând numărarea caracterelor, inserarea de octeţi şi umplerea cu biţi. Protocoalele legăturii de date pot oferi controlul erorilor pentru retransmiterea cadrelor distruse sau pierdute. Pentru a împiedica

un emiţător rapid să suprasolicite un receptor lent, protocolul legăturii de date poate realiza şi controlul fluxului. Mecanismul cu fereastră glisantă este foarte folosit pentru a integra controlul erorilor şi controlul fluxului într-un mod convenabil.

Protocoalele cu fereastră glisantă pot fi clasificate după dimensiunea ferestrei emiţătorului şi după dimensiunea ferestrei receptorului. Când ambele sunt egale cu 1, protocolul este pas-cu-pas (eng.: stop-and-wait). Când fereastra emiţătorului este mai mare ca 1, de exemplu pentru a împiedica emiţătorul să blocheze un circuit cu o întârziere mare de propagare, receptorul poate fi programat fie să elimine toate celelalte cadre cu excepţia următorului din secvenţă, fie să memoreze cadrele neordonate până când ele vor fi necesare.

În acest capitol au fost prezentate o serie de protocoale. Protocolul 1 a fost conceput pentru un mediu fără erori, în care receptorul poate face faţă oricărui flux de la emiţător. Protocolul 2 presupune existenţa unui mediu fără erori, dar introduce controlul fluxului. Protocolul 3 tratează problema erorilor prin utilizarea numerelor de secvenţă şi a algoritmului pas-cu-pas. Protocolul 4 permite comunicaţia bidirecţională şi introduce conceptul de ataşare (eng.: piggybacking). Protocolul 5 foloseşte un protocol cu fereastră glisantă şi revenire cu n paşi (eng.: go back n). Protocolul 6 foloseşte repetarea selectivă şi confirmări negative.

Protocoalele pot fi modelate folosind diferite tehnici ce ajută la demonstrarea corectitudinii lor (sau a lipsei acesteia). Modelele bazate pe automate finite şi modelele bazate pe reţele Petri sunt larg utilizate în acest scop.

Multe reţele folosesc la nivelul legătură de date unul dintre protocoalele orientate pe biţi – SDLC, HDLC, ADCCP sau LAPB. Toate aceste protocoale folosesc octeţi indicatori pentru delimitarea cadrelor şi inserarea de biţi pentru a preveni apariţia octeţilor indicatori în cadrul datelor. De asemenea toate aceste protocoale folosesc fereastra glisantă pentru controlul fluxului. Internet-ul foloseşte PPP ca principal protocol al legăturii de date pe liniile de tip punct-la-punct.

3.8 PROBLEME

1. Un mesaj de la un nivel mai înalt este spart în 10 cadre, fiecare dintre acestea având 80% şansă de a ajunge nemodificat. Dacă nu se face nici un control al erorilor de către protocolul legăturii de date, de câte ori va trebui transmis mesajul în medie pentru a-l obţine întreg la destinaţie?

2. Următoarea codificare a caracterelor este utilizată în cadrul unui protocol de nivel legătură de date:

A: 0100011 1; B: 11100011; FLAG: 01111110; ESC: 11100000

Determinaţi secvenţa de biţi transmisă (în binar) pentru cadrul format din următoarele 4 caractere: A B ESC FLAG, când fiecare din metodele de încadrare următoare sunt utilizate:

a) numărarea caracterelor
b) octeţi indicatori şi inserarea de octeţi
c) octeţi indicatori de început şi sfârşit, cu inserare de biţi.

3. Următorul fragment de date apare în mijlocul unui şir de date pentru care este folosit algoritmul de inserare de octeţi descris în text: A B ESC C ESC FLAG FLAG D. Care este ieşirea după inserare?

4. Unul dintre colegii Dvs. de clasă, Scrooge, a remarcat faptul că este ineficientă folosirea a 2 octeţi indicatori, unul pentru începutul cadrului, celălalt pentru sfârşit. Un singur octet indicator ar fi suficient, câştigându-se astfel un octet.

Sunteţi de acord?

5. Dacă în şirul de biţi 0111101111101111110 se inserează biţi, care este şirul de ieşire?

6. Când este utilizată inserarea de biţi, este posibil ca prin pierderea, inserarea sau modificarea unui singur bit să se provoace o eroare nedetectabilă prin suma de control? Dacă nu, de ce? Dacă da, de ce? Lungimea sumei de control joacă vreun rol aici?

7. Puteţi concepe o situaţie în care un protocol cu buclă deschisă (de exemplu un cod Hamming) poate fi preferabil protocoalelor cu buclă de reacţie (feedback), discutate pe parcursul acestui capitol?

8. Pentru a oferi o siguranţă mai mare decât cea pe care o poate da un singur bit de paritate, o schemă de codificare cu detecţie de erori foloseşte un bit de paritate pentru verificarea tuturor biţilor de ordin impar şi un al doilea bit de paritate pentru toţi biţii de ordin par. Care este distanţa Hamming pentru un astfel de cod?

9. Se transmit mesaje de 16 biţi folosind un cod Hamming. Câţi biţi de control sunt necesari pentru a asigura detectarea şi corectarea de către receptor a erorilor de un bit? Prezentaţi secvenţa de biţi transmisă pentru mesajul 1101001100110101. Presupuneţi că se foloseşte paritare pară.

10. Un octet (8 biţi) cu valoarea binară 10101111 trebuie codificat utilizând un cod Hamming cu paritate pară. Care este valoarea binară după codificare?

11. Un cod Hamming de 12 biţi a cărui valoare în hexazecimal este 0xE4F soseşte la receptor. Care este valoarea hexazecimală originală? Presupuneţi ca maxim un bit este eronat.

12. Un mod de a detecta erorile este de a transmite datele ca un bloc de n rânduri a câte k biţi pe rând şi adăugarea de biţi de paritate pentru fiecare rând şi fiecare coloană. În colţul din dreapta jos este bitul de paritate care verifică linia şi coloana sa. Va detecta această schemă toate erorile singulare? Dar erorile duble? Dar erorile triple?

13. Un bloc de biţi cu *n* rânduri şi *k* coloane foloseşte biţi de paritate verticală şi orizontală pentru detecţia erorilor, Să presupunem că datorită erorilor de transmisie sunt inversaţi exact 4 biţi. Deduceţi o expresie pentru exprimarea probabilităţii ca eroarea să nu fie detectată.

14. Ce rest se obţine prin împărţirea lui x^7+x5+1 la polinomul generator x3+1?

15. Secvenţa de biţi 10011101 este transmisă folosind metoda CRC descrisă anterior. Polinomul generator este x3+1. Prezentaţi secvenţa de biţi transmisă. Se presupune că al treilea de la stânga este inversat în timpul transmisiei. Arătaţi că această eroare este detectată de receptor.

16. Protocoalele legăturii de date pun aproape întotdeauna CRC-ul în partea finală

şi nu în antet. De ce?

17. Un canal are o rată de transmisie a biţilor de 4 Kbps şi o întârziere de propagare de 20 ms. Pentru ce domeniu al dimensiunii cadrelor metoda pas-cu-pas (stop-and-wait) are o eficienţă de cel puţin 50%? ††††

19. În protocolul 3, este posibil ca emiţătorul să pornească contorul de timp, atunci când acesta merge deja? Dacă da, când se poate întâmpla acest lucru? Daca nu, de ce este imposibil?

20. Imaginaţi un protocol cu fereastră glisantă ce foloseşte suficienţi biţi pentru numerele de secvenţă, astfel încât să nu apară niciodată suprapuneri. Ce relaţie trebuie să existe între cele patru limite ale ferestrelor şi dimensiunea ferestrei?

21. Dacă în procedura *between* din protocolul 5 este verificată condiţia a < b < c în locul condiţiei a < b < c, ar avea aceasta vreun efect asupra corectitudinii protocolului sau eficienţei sale? Explicaţi răspunsul.

22. În protocolul 6, când soseşte un cadru de date, este făcută o verificare pentru a se vedea dacă numărul de secvenţă diferă de cel aşteptat şi *no nak* este adevărat. Dacă ambele condiţii sunt îndeplinite, este trimis un NAK. Altfel, este pornit contorul de timp auxiliar. Presupuneţi că ar fi omisă clauza else. Ar afecta aceasta corectitudinea protocolului?

23. Presupunem că bucla while cu trei instrucţiuni din finalul protocolului 6 a fost ştearsă din cod. Ar afecta aceasta corectitudinea protocolului sau doar performanţa? Explicaţi răspunsul.

24. Presupunem că instrucţiunea *case* pentru erorile de sumă de control a fost scoasă din instrucţiunea *switch* din protocolul 6. Cum ar afecta aceasta operarea protocolului?

25. În protocolul 6 codul pentru *frame_arrival* are o secţiune folosită pentru NAK-uri. Această secţiune este invocată în cazul în care cadrul sosit este un NAK şi este îndeplinită încă o condiţie. Indicaţi un scenariu în care prezenţa acestei condiţii este esenţială.

26. Imaginaţi-vă că scrieţi un program la nivelul legătură de date pentru o linie folosită pentru a primi date, dar nu şi pentru a trimite. Celălalt capăt foloseşte HDLC, cu un număr de secvenţă pe 3 biţi şi o dimensiune a ferestrei de 7 cadre. Aţi dori să memoraţi cât mai multe cadre din secvenţă pentru a creşte eficienţa, dar nu vă este permis să modificaţi programul transmiţătorului. Este posibil să aveţi o fereastră la receptor mai mare ca 1 şi totuşi să existe garanţia că protocolul nu va eşua? Dacă da, care este fereastra cea mai mare care poate fi utilizată în siguranţă?

---

††††Un trunchi T1 lung de 3000 km este folosit pentru a transmite cadre de 64 de biţi folosind protocolul 5. Dacă viteza de propagare este de 6 psec/km, pe câţi biţi trebuie reprezentate numerele de secvenţă?

27. Consideraţi operarea protocolului 6 pe o linie fără erori de 1 Mbps. Dimensiunea maximă a cadrului este 1000 biţi. Pachetele noi sunt generate la un interval de aproape o secundă. Intervalul de expirare a timpului este de 10 ms. Dacă ar fi eliminate confirmările speciale pentru contorul de timp, ar putea apărea expirări de timp inutile. De câte ori ar trebui transmis în medie un mesaj?

28. În protocolul 6, $MAX\_SEQ = 2^n-1$. Deşi această condiţie este evident necesară pentru a utiliza eficient biţii din antet, nu s-a demonstrat că ea este şi esenţială. Ar funcţiona protocolul corect pentru $MAX\_SEQ = 4$ de exemplu?

29. Cadrele de 1000 de biţi sunt transmise pe un canal printr-un satelit geostaţionar de 1 Mbps a cărui întârziere de propagare de la Pământ este de 270 milisecunde. Confirmările sunt întotdeauna ataşate cadrelor de date. Antetele sunt foarte scurte. Sunt folosite numere de secvenţă pe 3 biţi. Care este utilizarea maximă realizabilă a canalului pentru:

a) Pas-cu-pas (stop-and-wait);
b) Protocolul 5;
c) Protocolul 6.

30. Calculaţi fracţiunea din lărgimea de bandă ce este pierdută datorită supraîncărcării (antete şi retransmisie) pentru protocolul 6 pe un canal de satelit de 50 Kbps, foarte încărcat cu cadre de date constând din 40 de biţi antet şi 3960 biţi de date. Presupuneţi o întârziere de la Pământ la satelit de 270 milisecunde. Cadrele ACK nu apar niciodată. Cadrele NAK sunt de 40 de biţi. Rata de erori pentru cadrele de date este de 1% şi rata de erori pentru cadrele NAK este neglijabilă. Numerele de secvenţă sunt pe 8 biţi.

31. Se consideră un canal prin satelit fără erori, de 64 Kbps, folosit pentru a transmite cadre de date de 512 octeţi într-o singură direcţie, cu confirmări foarte scurte ce se întorc pe cealaltă cale. Care este productivitatea maximă pentru dimensiuni ale ferestrei de 1, 7, 15 şi 127? Presupuneţi o întârziere de la Pământ la satelit de 270 milisecunde.

32. Un cablu lung de 100 km funcţionează la rata de transmisie de date T1. Viteza de propagare pe cablu este 2/3 din viteza luminii. Câţi biţi încap pe cablu?

33. Se presupune că se modelează protocolul 4 utilizând modelul automatelor finite. Câte stări există pentru fiecare maşină? Câte stări există pentru canalul de comunicaţie? Dar pentru un sistem complet (două maşini şi canalul)? Se ignoră erorile de sumă de control.

34. Determinaţi o secvenţă executabilă pentru reţeaua Petri din fig. 3-23 corespunzătoare secvenţei de stări (000), (01A), (01-), (010), (01A) în fig. 3-21. Explicaţi în cuvinte ce reprezintă secvenţa respectivă.

35. Date fiind regulile de tranziţie AC^B, B^AC, CD^E şi E^CD, desenaţi reţeaua Petri descrisă de ele. Folosind reţeaua Petri, desenaţi graful finit al stărilor

accesibile din starea inițială ACD. Ce concept bine-cunoscut din științа calculatoarelor folosește acest model de reguli de tranziție?

36. PPP se bazează pe HDLC, care folosește inserarea de biți pentru a împiedica octeții indicatori accidentali din interiorul informației utile să provoace confuzii. Dați cel puțin un motiv pentru care PPP folosește în locul acesteia inserarea de octeți.

37. Care este supraîncărcarea minimă în transmiterea unui pachet IP folosind PPP? Luați în considerare doar supraîncărcarea introdusă de PPP însuși, nu și supraîncărcarea produsă de antetul IP.

38. Scopul acestui exercițiu este implementarea unui mecanism de detectare a erorilor folosind algoritmul standard CRC prezentat în text. Scrieți două programe, *generator* și *verificator*. Programul generator citește de la intrarea standard mesaje de n biți ca șiruri de 1 și 0, ca o linie de text ASCII. A doua linie este polinomul generator pe k biți, citit tot ca text ASCII. La ieșire, programul va afișa la ieșirea standard (standard output) o linie de text ASCII cu $n+k$ caractere 0 și 1, reprezentând mesajul de transmis. Apoi afișează polinomul, așa cum l-a citit. Programul verificator citește ieșirea programului generator și afișează un mesaj care indică dacă aceasta este corectă sau nu. Se va scrie apoi un program, *alterează,* care inversează un bit din prima linie depinzând de unul din parametri cu care a fost apelat (bitul cel mai din stânga se consideră bitul 1), dar copiază restul corect. Tastând:

generator < fișier | verificator

ar trebui să vedeți că mesajul este corect, dar tasând:

generator < fișier | alterează argument | verifica

ar trebui să primiți un mesaj de eroare.

39. Scrieți un program care să simuleze comportamentul unei rețele Petri. Programul trebuie să citească regulile de tranziție și o listă de stări corespunzând nivelului legătură al rețelei ce emite un nou pachet sau acceptă un pachet. Din starea inițială, de asemenea citită de pe mediul de intrare, programul trebuie să aleagă tranzițiile permise și să le execute aleator, verificând dacă un calculator gazdă acceptă două mesaje fără ca un alt calculator gazdă să emită unul nou între ele.

SUBNIVELUL DE ACCES LA MEDIU

Aşa cum am arătat în Cap. 1, reţelele pot fi împărţite în două categorii: cele care utilizează conexiuni punct-la-punct şi cele care utilizează canale cu difuzare (broadcast channels). Acest capitol se ocupă de reţelele cu difuzare (broadcast networks) şi de protocoalele lor.

În orice reţea cu difuzare, una dintre probleme este determinarea utilizatorului cu drept de acces la canal în cazul în care există mai mulţi utilizatori concurenţi. Pentru a lămuri lucrurile, să considerăm o teleconferinţă în care şase persoane, vorbind de la şase telefoane diferite, sunt conectate astfel încât fiecare îi poate auzi pe ceilalţi şi poate vorbi cu ei. Este foarte probabil ca atunci când cineva se opreşte din vorbit, doi sau mai mulţi să înceapă să vorbească simultan, ceea ce va duce la haos. Într-o întâlnire faţă-în-faţă, haosul este evitat prin mijloace externe – de exemplu, prin ridicarea mâinii pentru a cere permisiunea de a vorbi. Când este disponibil un singur canal de comunicaţie, este mult mai greu să determini cine urmează să ia cuvântul. Sunt cunoscute multe protocoale de rezolvare a acestei probleme şi ele constituie conţinutul acestui capitol. În literatura de specialitate, canalele cu difuzare sunt uneori numite **canale multiacces** (multiaccess channels), sau **canale cu acces aleator** (random access channels).

Protocoalele folosite pentru a determina cine urmează într-un canal multiacces aparţin unui subnivel al nivelului legătură de date, numit subnivelul **MAC** (**Medium Access Control**, rom: controlul accesului la mediu). Subnivelul MAC este important mai ales pentru reţelele de tip LAN – Local Area Network (le vom numi prescurtat LAN-uri), care utilizează aproape toate un canal multiacces ca bază pentru comunicaţie. Din contră, reţelele de tip WAN – Wide Area Network (le vom numi

WAN-uri) utilizează legături punct-la-punct, cu excepţia reţelelor prin satelit. Datorită faptului că LAN-urile şi canalele multiacces sunt atât de strâns legate, în acest capitol vom discuta la modul general atât despre LAN-uri cât şi despre reţele prin satelit şi alte reţele cu difuzare. Deoarece canalele multiaccess şi reţelele locale sunt subiecte atât de apropiate, în acest capitol vom discuta mai multe subiecte legate de reţelele locale în general, incluzând şi subiecte care nu sunt strict specifice subnivelului MAC.

Tehnic vorbind, subnivelul MAC reprezintă partea de jos a nivelului legătură de date, deci logic ar fi fost să îl fi studiat înainte de a trece în revistă toate protocoalele punct-la-punct din cap. 3. Dar, pentru majoritatea oamenilor, înţelegerea protocoalelor care implică mai multe părţi este mai uşoară după ce au înţeles bine protocoalele care implică numai două părţi. Pentru acest motiv am deviat puţin de la stilul de prezentare strict ascendent al ierarhiei reţelelor.

4.1 PROBLEMA ALOCĂRII CANALULUI

Tema centrală a acestui capitol o reprezintă modul de alocare a unui singur canal cu difuzare între mai mulţi utilizatori concurenţi. Mai întâi vom arunca o privire de ansamblu asupra schemelor statice şi dinamice de alocare. Apoi vom studia câţiva algoritmi specifici.

4.1.1 Alocarea statică a canalului în reţelele LAN şi MAN

Modul tradiţional de alocare a unui singur canal, cum ar fi cablul telefonic, între mai multi utilizatori concurenţi este multiplexarea cu diviziunea frecvenţei (FDM – Frequency Division Multiplexing). Dacă există N utilizatori, banda de transmisie este împărţită în N părţi egale (vezi fig. 2-24), fiecărui utilizator fiindu-i alocată una dintre acestea. Deoarece fiecare utilizator are o bandă de frecvenţă proprie, nu există interferenţe între utilizatori. Atunci când există doar un număr mic şi constant de utilizatori, fiecare având un trafic încărcat (şi bazat pe utilizarea zonelor tampon), cum ar fi, de exemplu, oficiile de comutare ale companiilor de telecomunicaţie, FDM este un mecanism de alocare simplu şi eficient.

Cu toate acestea, atunci când numărul emiţătorilor este mare şi variază în permanenţă, sau când traficul este de tip rafală, FDM prezintă câteva probleme. Dacă spectrul benzii este împărţit în N regiuni şi sunt mai puţin de N utilizatori care vor să comunice, o bună parte din bandă se va risipi. Dacă sunt mai mult de N utilizatori care vor să comunice, unii dintre ei nu o vor putea face, din lipsă de spaţiu în banda de transmisie, chiar dacă există utilizatori care au primit câte o parte din bandă şi transmit sau recepţionează mesaje extrem de rar.

Chiar dacă presupunem că numărul utilizatorilor ar putea fi menţinut în vreun fel constant la valoarea N, divizarea singurului canal disponibil în subcanale statice este, evident, ineficientă. Principala problemă este că atunci când unii utilizatori sunt inactivi, bucata lor de bandă se pierde pur şi simplu. Ei nu o folosesc, dar nici alţii nu au voie să o utilizeze. Mai mult, în majoritatea sistemelor de calcul, traficul

de date este extrem de diferenţiat (sunt uzuale raporturi de 1000:1 între traficul de vârf şi cel mediu). În consecinţă, majoritatea canalelor vor fi libere în cea mai mare parte a timpului.

Performanţele slabe ale alocării FDM statice pot fi uşor observate dintr-un simplu calcul făcut cu ajutorul teoriei cozilor. Să luăm, pentru început, întârzierea medie, *T,* pentru un canal cu capacitatea *C* bps, la o rată a sosirilor de Z cadre/sec. Fiecare cadru are o lungime dată de o funcţie de densitate de probabilitate exponenţială cu media de 1/p biţi/cadru. Cu aceşti parametri, rata sosirilor este *Z* cadre/sec, iar viteza de servire este pC cadre-sec. Din teoria cozilor poate fi demonstrat că pentru timpi Poisson de sosire şi de servire, vom avea

$$\frac{1}{pC-X}$$

De exemplu, dacă C este 100 Mbps, lungimea medie a cadrului, 1/p, este de 10.000 de biţi, iar rata sosirilor, X, este 5.000 cadre/sec, atunci T = 200 ps. Trebuie remarcat că dacă ignoram întârzierea dată de teoria cozilor şi am fi vrut să determinăm doar cât timp va dura să trimitem un cadru de 10.000 de biţi într-o reţea de 100 Mbps, am fi obţinut răspunsul (incorect) de 100 ps. Rezultatul, astfel calculat, este corect doar când nu există competiţie pentru canal.

Acum să divizăm canalul în *N* subcanale independente, fiecare cu o capacitate de *C/N* bps. Rata medie a intrărilor pe fiecare subcanal va fi acum de X/*N*. Recalculând *T* vom obţine:

$$T = \frac{1}{p\,(C/N) - (X/N)\,pC - X} = \frac{N}{pC - X} = NT \qquad (4\text{-}$$

Întârzierea medie la FDM este de *N* ori mai mare decât în cazul în care toate cadrele ar fi fost, printr-o scamatorie, aranjate în ordine într-o mare coadă centrală.

Exact aceeaşi logică utilizată la FDM se poate aplica şi la multiplexarea cu diviziunea timpului (TDM – Time Division Multiplexing). Fiecărui utilizator îi este alocată static fiecare a *N*-a cuantă. Dacă un utilizator nu îşi foloseşte timpul alocat, acesta rămâne nefolosit. Acelaşi lucru se întâmplă şi dacă divizăm reţelele în mod fizic. Revenind la exemplul anterior, dacă am înlocui reţeaua de 100 Mbps cu zece reţele de 10 Mbps fiecare şi dacă am aloca static fiecăreia câte un utilizator, întârzierea medie ar sări de la 200 ps la 2 ms.

Deoarece nici una dintre metodele statice de alocare a canalului nu funcţionează bine în condiţii de trafic în rafală, vom studia în continuare metodele dinamice.

4.1.2 Alocarea dinamică a canalului în reţelele LAN şi MAN

Înainte de a începe prezentarea numeroaselor metode de alocare a canalului, care fac obiectul acestui subcapitol, merită să formulăm cu atenţie problema alocării. La baza întregii activităţi din acest domeniu stau câteva ipoteze-cheie, descrise în continuare.

1. **Modelul staţiilor**. Acest model constă din *N* **staţii** independente (calculatoare, telefoane, dispozitive de comunicare personală etc.), fiecare

având un program sau un utilizator care generează cadre de transmis. Staţiile sunt uneori denumite terminale. Probabilitatea de generare a unui cadru într-un interval de lungime Ni este XNt, unde X este o constantă (rata sosirilor de cadre noi). Odată ce a fost generat un cadru, staţia se blochează şi nu mai face nimic până la transmiterea cu succes a cadrului.

2. **Ipoteza canalului unic**. Există un singur canal accesibil pentru toate comunicaţiile. Toate staţiile pot transmite prin el şi pot recepţiona de la el. În ceea ce priveşte partea de hardware, toate staţiile sunt echivalente, deşi protocolul software le poate acorda priorităţi diferite.
3. **Ipoteza coliziunii**. Dacă două cadre sunt transmise simultan, ele se suprapun, iar semnalul rezultat va fi neinteligibil. Acest eveniment se numeşte **coliziune**. Toate staţiile pot detecta coliziuni. Un cadru care a intrat în coliziune cu un alt cadru trebuie retransmis ulterior. Nu există alte erori în afara celor generate de coliziuni.
4. **Timp continuu**. Transmisia cadrelor poate surveni în orice moment. Nu există un ceas comun, care să împartă timpul în intervale discrete.
5. **Timp discret**. Timpul este împărţit în intervale discrete (cuante). Transmisia cadrelor porneşte întotdeauna la începutul unei cuante. O cuantă poate conţine 0, 1, sau mai multe cadre, corespunzător unei cuante de aşteptare, unei transmisiuni efectuate cu succes sau, respectiv, unei coliziuni.
6. **Detecţia purtătoarei**. Staţiile pot afla dacă un canal este liber sau nu înainte de a încerca să-l utilizeze. Dacă el este deja ocupat, nici o staţie nu va mai încerca să îl utilizeze până când nu se va elibera.
7. **Nedetecţia purtătoarei**. Staţiile nu pot afla starea canalului înainte de a încerca să îl utilizeze. Ele pur şi simplu încep să transmită. Abia după aceea vor putea determina dacă transmisia s-a efectuat cu succes sau nu.

Este momentul să discutăm puţin despre aceste ipoteze. Prima dintre ele spune că staţiile sunt independente, iar cadrele sunt generate cu o frecvenţa constantă. De asemenea, se presupune implicit că transmisia fiecărei staţii este controlată de un singur program sau de un singur utilizator, deci atâta timp cât staţia este blocată, ea nu va genera noi cadre. Modelele mai sofisticate permit existenţa staţiilor multiprogramate, care pot genera noi cadre în timp ce staţia este blocată, dar analiza acestor staţii este mult mai complexă.

Ipoteza canalului unic este de fapt inima problemei. Nu există mijloace externe de comunicare. Staţiile nu pot ridica mâinile pentru a cere profesorului permisiunea de a vorbi.

Ipoteza coliziunii este, de asemenea, o ipoteză de bază, deşi în unele sisteme (între care remarcăm sistemele cu spectru larg de transmisie) ea este relaxată, cu rezultate surprinzătoare. De asemenea, unele LAN-uri, cum ar fi cele de tip token-ring, utilizează un mecanism de eliminare a conflictelor, care elimină coliziunile.

Există două ipoteze alternative despre timp. Într-una din ele timpul este continuu, iar în cealaltă este discret. Unele sisteme consideră timpul într-un fel, altele în celălalt fel, aşa că le vom discuta şi analiza pe amândouă. Evident, pentru un sistem dat, numai una dintre ipoteze este valabilă.

În mod similar, o reţea poate avea sau nu facilităţi de detecţie a purtătoarei.

Reţelele LAN au în general detecţie de purtătoare, dar reţelele prin satelit nu (datorită întârzierii mari de propagare). Staţiile din reţelele cu detecţie de purtătoare îşi pot termina transmisia prematur, dacă descoperă că au intrat în coliziune cu o altă transmisie. De notat că aici înţelesul cuvântului „purtătoare” se referă la semnalul electric de pe cablu şi nu are nimic de a face cu vreun alt tip de purtătoare.

4.2 PROTOCOALE CU ACCES MULTIPLU

Sunt cunoscuţi mulţi algoritmi de alocare a unui canal cu acces multiplu. În secţiunile care urmează vom studia un eşantion reprezentativ al celor mai interesanţi algoritmi şi vom da exemple de utilizare a lor.

4.2.1 ALOHA

În anii '70, Norman Abramson şi colegii săi de la Universitatea din Hawaii au elaborat o nouă şi elegantă metodă de rezolvare a problemei alocării canalului. De atunci, munca lor a fost continuată de mulţi cercetători (Abramson, 1985). Deşi realizarea lui Abramson, numită sistemul ALOHA,

utiliza difuzarea prin radio de la sol, ideea de bază se poate aplica la orice sistem în care utilizatori ce nu pot fi localizaţi concurează la utilizarea unui unic canal partajat.

Vom discuta două versiuni ale protocolului ALOHA: ALOHA pur şi ALOHA cuantificat. Ele diferă prin faptul că timpul este sau nu divizat în intervale discrete, în care trebuie să se potrivească orice cadru. ALOHA pur nu cere sincronizare de timp globală, pe când ALOHA cuantificat cere.

ALOHA pur

Ideea de bază într-un sistem ALOHA este simplă: utilizatorii sunt lăsaţi să transmită ori de câte ori au date de trimis. Bineînţeles că vor exista coliziuni, iar cadrele intrate în coliziune vor fi distruse. Oricum, datorită proprietăţii de reacţie a difuzării, un emiţător poate afla oricând dacă mesajul său a fost distrus, ascultând canalul, la fel ca şi ceilalţi utilizatori. Într-o reţea LAN, reacţia este imediată; într-o reţea prin satelit, există o întârziere de 270 ms înainte ca emiţătorul să afle dacă transmisia s-a încheiat cu succes. În cazul în care cadrul trimis a fost distrus, emiţătorul aşteaptă un interval oarecare de timp şi îl trimite din nou. Timpul de aşteptare trebuie să fie aleatoriu, altfel aceleaşi cadre vor intra în coliziune iar şi iar, blocându-se reciproc la nesfârşit. Sistemele în care mai mulţi utilizatori partajează un canal comun într-un mod care poate duce la conflicte sunt cunoscute sub numele de **sisteme cu conflicte** (**contention systems**).

În fig. 4-l este prezentată o schiţă de generare a cadrelor într-un sistem ALOHA. Am ales să reprezentăm cadre de aceeaşi lungime, pentru că productivitatea sistemelor ALOHA este maximizată în cazul în care avem cadre de lungime uniformă, faţă de cazul în care avem cadre de lungime variabilă.

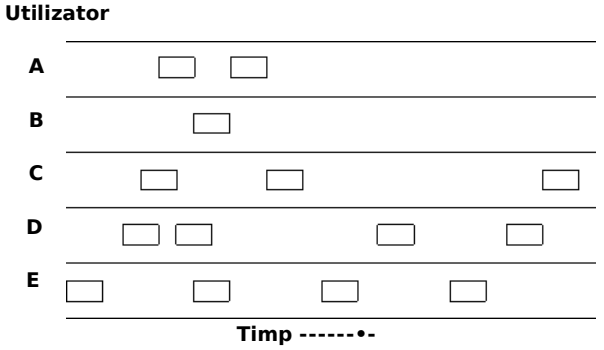


Fig. 4-1. În ALOHA pur, cadrele sunt transmise la momente complet arbitrare.

Ori de câte ori două cadre încearcă să ocupe canalul în acelaşi timp, se va produce o coliziune şi amândouă vor fi denaturate. Dacă primul bit al unui nou cadru se suprapune cu ultimul bit al unui cadru aproape terminat, amândouă cadrele vor fi total distruse şi amândouă vor trebui retransmise mai târziu. Suma de control nu poate (şi nu trebuie) să distingă între o pierdere totală şi o ratare „la mustaţă”. Ceea ce este rău este rău.

O întrebare foarte interesantă este: care este eficienţa unui canal ALOHA? Cu alte cuvinte, ce fracţiune din cadrele transmise nu intră în coliziune în aceste

circumstanţe haotice? Să considerăm mai întâi o colectivitate infinită de utilizatori interactivi stând în faţa calculatoarelor (staţiilor) lor. Un utilizator este întotdeauna într-una din cele două stări: introduce caractere sau aşteaptă. Iniţial, toţi utilizatorii sunt în prima stare, scriind. Când termină o linie, utilizatorul se opreşte din scris, aşteptând un răspuns. Atunci staţia transmite pe canal un cadru conţinând linia şi verifică dacă transmisia s-a efectuat cu succes. Dacă da, utilizatorul vede răspunsul şi se apucă din nou de scris. Dacă nu, utilizatorul continuă să aştepte, iar cadrul va fi transmis în mod repetat, până când transmisia se va încheia cu succes.

Să numim „interval de cadru" timpul necesar pentru a transmite un cadru standard, de lungime fixă (adică lungimea cadrului împărţită la rata biţilor). Vom presupune că populaţia infinită de utilizatori generează cadre noi conform unei distribuţii Poisson cu media de N cadre pe interval de cadru (ipoteza populaţiei infinite este necesară pentru a ne asigura că N nu descreşte pe măsură ce utilizatorii se blochează). Dacă $N > 1$, utilizatorii generează cadre cu o rată mai mare decât capacitatea de transmisie a canalului şi aproape fiecare cadru va suferi o coliziune. Pentru o productivitate rezonabilă ar trebui ca $0 < N < 1$.

În plus faţă de noile cadre, staţiile mai generează şi copii ale cadrelor care au suferit anterior coliziuni. Să presupunem în continuare că probabilitatea de a avea k încercări de transmisie pe interval de cadru, incluzând şi retransmisiile, are de asemenea o distribuţie Poisson, cu media G pe interval de cadru. Evident, $G > N$. La încărcare redusă (adică N ® 0), vor fi puţine coliziuni, deci puţine retransmisii, aşa că G ® N. La încărcare mare vor fi multe coliziuni, deci $G > N$. Orice încărcare am avea, productivitatea este chiar încărcarea dată, G, înmulţită cu probabilitatea ca o transmisie să se încheie cu succes – adică $S = GP_0$, unde P_0 este probabilitatea ca un cadru să nu sufere coliziuni.

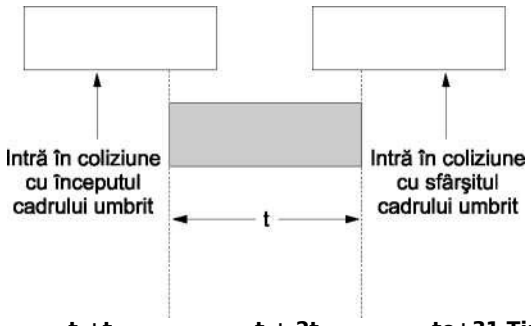


Fig. 4-2. Perioada vulnerabilă pentru cadrul umbrit.

Un cadru nu va suferi coliziuni dacă nici un alt cadru nu va fi emis în intervalul de un cadru socotit de la începutul lui, aşa cum se arată în fig. 4-2. În ce condiţii cadrul umbrit va ajunge întreg? Fie t timpul necesar emisiei unui cadru. Dacă un alt utilizator a generat un cadru între t_0 şi $t_0 + t$, sfârşitul acelui cadru va intra în coliziune cu începutul cadrului umbrit. De fapt, soarta cadrului umbrit era deja pecetluită chiar înainte de transmisia primului bit, dar cum în ALOHA pur, o staţie nu ascultă canalul înainte de transmisie, nu are cum să ştie că un alt cadru se află

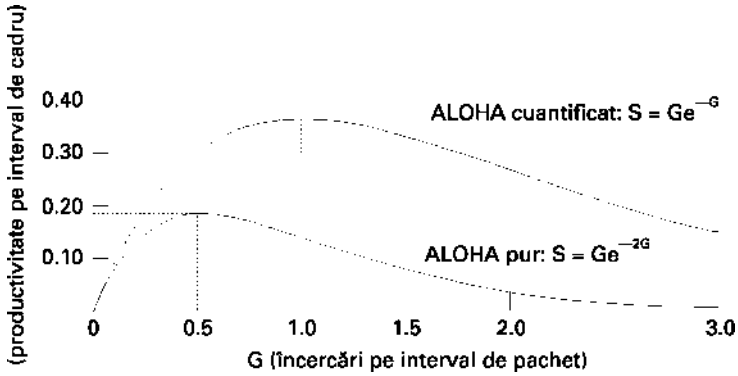
deja în curs de transmisie. Similar, orice alt cadru care începe între $t_0 + t$ şi $t_0 + 2t$ va nimeri peste sfârşitul cadrului umbrit.

Probabilitatea ca într-un interval de cadru dat să fie generate un număr k de cadre este modelată de distribuţia Poisson:

$$\Pr[k] = \frac{G^k e^{-G}}{k!} \quad (4\text{-}2)$$

deci probabilitatea generării a zero cadre este doar e^{-G}. Într-o perioadă de timp cât două intervale de cadru, media numărului de cadre generate este $2G$. Astfel, probabilitatea ca nici o transmisie să nu înceapă în timpul perioadei de timp vulnerabile este dată de $P_0 = e^{-2G}$.
Luând $S = GP_0$, obţinem:

$$S = Ge^{-2G}$$



c/>

Fig. 4-3. Productivitatea în funcţie de traficul oferit pentru sistemele ALOHA.

Relaţia dintre traficul oferit şi productivitate este prezentată în fig. 4-3. Productivitatea maximă este obţinută la $G = 0.5$, cu $S = 1/2e$, adică aproximativ 0.184. Cu alte cuvinte, cea mai bună performanţă la care putem spera este o utilizare a canalului de 18 procente. Acest rezultat nu este prea încurajator, dar, în situaţia aceasta în care fiecare utilizator transmite la dorinţă, cu greu ne-am fi putut aştepta la o performanţă de sută la sută.

ALOHA cuantificat

În 1972, Roberts a publicat o metodă de dublare a capacităţii unui sistem ALOHA (Roberts, 1972). Propunerea lui era să se împartă timpul în intervale discrete, fiecare interval corespunzând unui cadru. Această abordare cere ca utilizatorii să cadă de acord asupra mărimii cuantelor. O cale de a obţine sincronizarea ar fi ca o staţie specială să emită un „bip” la începutul fiecărui interval, ca un tact de ceas.

În metoda lui Roberts, care a devenit cunoscută sub numele de **ALOHA cuantificat** (**slotted ALOHA**), în contrast cu metoda lui Abramson – **ALOHA pur** (**pure ALOHA**), unui calculator nu îi este permis să emită ori de câte ori este apăsată tasta „Return”. El este nevoit să aştepte începutul următoarei cuante. Astfel, protocolul ALOHA pur este transformat din continuu în discret. Deoarece acum perioada vulnerabilă este înjumătăţită, probabilitatea ca în intervalul cadrului nostru de test să nu mai apară un alt trafic este e^{-G}, ceea ce conduce la:

$$S = Ge \quad (4\text{-}3)$$

Aşa cum reiese din fig. 4-3, ALOHA cuantificat prezintă un maxim la $G = 1$, cu o productivitate de $S = 1/e$, adică aproximativ 0.368, dublu faţă de ALOHA pur. Dacă sistemul operează la $G = 1$, probabilitatea unei cuante neutilizate este 0.368 (din ecuaţia 4-2). Cea mai bună performanţă la care ne putem aştepta de la ALOHA cuantificat este: 37% din cuante neutilizate, 37% cadre transmise cu succes şi 26% coliziuni. Lucrul cu valori mai mari ale lui G reduce numărul cuantelor neutilizate, dar îl măreşte exponenţial pe cel al coliziunilor. Pentru a vedea cum se explică rapida creştere a numărului coliziunilor odată cu G, să considerăm transmisia unui cadru de test. Probabilitatea ca el să evite o coliziune este e^{-G}, adică probabilitatea ca toţi ceilalţi utilizatori să nu transmită în acest interval. Probabilitatea unei coliziuni este deci $1 - e^{-G}$. Probabilitatea ca o transmisie să se efectueze exact din k încercări (adică după $k - 1$ coliziuni, urmate de un succes) este

$$P_k = e^{-G}(1 - e^{-G})^{k-1}$$

Numărul de transmisii prognozat pentru fiecare apăsare a tastei „Return”, E, este deci

$$E = \sum_{k=1}^{\infty} kP_t = \sum_{k=1}^{\infty} ke^{-G}(1 - e^{-G})^{k-1} = e^{G}$$

Ca urmare a dependenţei exponenţiale a lui E faţă de G, creşteri mici ale încărcării canalului pot reduce drastic performanţele sale.

ALOHA cuantificat este important pentru un motiv care poate nu este evident la prima vedere. A fost conceput în anii '70, a fost folosit în câteva sisteme experimente timpurii, apoi a fost aproape uitat. Când a fost inventat accesul la Internet prin cablu a apărut dintr-o dată problema alocării unui singur canal între utilizatori multipli aflaţi în concurenţă – astfel încât ALOHA cuantificat a fost scos de la naftalină pentru a salva situaţia. S-a întâmplat frecvent ca protocoale perfect valide să fie date uitării din motive politice (de exemplu, deoarece o mare companie vrea ca toată lumea să facă lucrurile aşa cum zice ea), dar la ani distanţă o persoană inteligentă descoperă că un protocol de mult uitat rezolva o anumită problemă curentă. Din acest motiv, în acest capitol vom studia un număr de protocoale elegante care nu sunt actualmente folosite la scară largă, dar care ar putea foarte simplu să fie utile în aplicaţiile viitorului, dacă suficienţi ingineri de reţea sunt conştienţi de existenţa lor. Desigur, vom studia mai multe protocoale aflate în folosinţă curentă.

4.2.2 Protocoale cu acces multiplu şi detecţie de purtătoare

Cu ALOHA cuantificat se poate atinge un grad de utilizare a canalului de până la $1/e$. Acest lucru nu este surprinzător, dacă ne gândim că staţiile transmit când doresc, fără a fi atente la ceea ce fac celelalte staţii şi, în consecinţă, vor exista numeroase coliziuni. Oricum, în reţelele locale, staţiile pot detecta ce fac celelalte staţii şi îşi pot adapta comportamentul în mod corespunzător. Astfel de reţele pot obţine un grad de utilizare mult mai bună decât $1/e$. În această secţiune vom discuta câteva protocoale pentru îmbunătăţirea a performanţei.

Protocoalele în care staţiile ascultă pentru a detecta o purtătoare (adică o

transmisie) şi acţionează corespunzător se numesc **protocoale cu detecţie de purtătoare** (**carrier sense protocols**). Kleinrock şi Tobagi (1975) au analizat în detaliu câteva protocoale de acest tip. În continuare vom prezenta câteva versiuni ale protocoalelor cu detecţie de purtătoare.

CSMA persistent şi nepersistent

Primul protocol cu detecţie de purtătoare pe care îl vom studia în acest material se numeşte **CSMA l-persistent** (**Carrier Sense Multiple Access**, rom: acces multiplu cu detecţie de purtătoare). Atunci când o staţie are date de transmis, mai întâi ascultă canalul pentru a vedea dacă nu cumva transmite altcineva în acel moment. În cazul în care canalul este ocupat, staţia aşteaptă până la eliberarea sa. Atunci când staţia detectează canalul liber, transmite un cadru. Dacă se produce o coliziune, staţia aşteaptă o perioadă aleatorie de timp şi o ia de la început. Protocolul se cheamă l-persistent, pentru că probabilitatea ca o staţie să transmită atunci când găseşte canalul liber este egală cu 1.

Întârzierea de propagare are o influenţă importantă asupra performanţei protocolului. Există o oarecare şansă ca, imediat după ce o staţie începe să transmită, o altă staţie să devină pregătită de transmisie şi să asculte canalul. Dacă semnalul primei staţii nu a ajuns încă la cea de-a doua, aceasta din urmă va detecta canalul liber şi va începe la rândul ei să emită, rezultând o coliziune. Cu cât este mai mare întârzierea de propagare, cu atât acest efect devine mai important, iar performanţa protocolului scade.

Chiar dacă întârzierea de propagare ar fi zero, tot s-ar mai produce coliziuni. Dacă două staţii devin gata de transmisie în timpul transmisiunii unei a treia staţii, amândouă vor aştepta politicos până la sfârşitul ei, după care vor începe să transmită simultan, producându-se o coliziune. Dacă ele nu ar fi atât de nerăbdătoare, s-ar produce mai puţine coliziuni. Chiar şi aşa, acest protocol este semnificativ mai bun decât ALOHA pur, întrucât ambele staţii au bunul simţ să nu interfereze cu cadrul celei de-a treia staţii. Intuitiv, acest fapt va conduce la o performanţă mai bună decât ALOHA pur. Acelaşi lucru este valabil şi pentru ALOHA cuantificat.

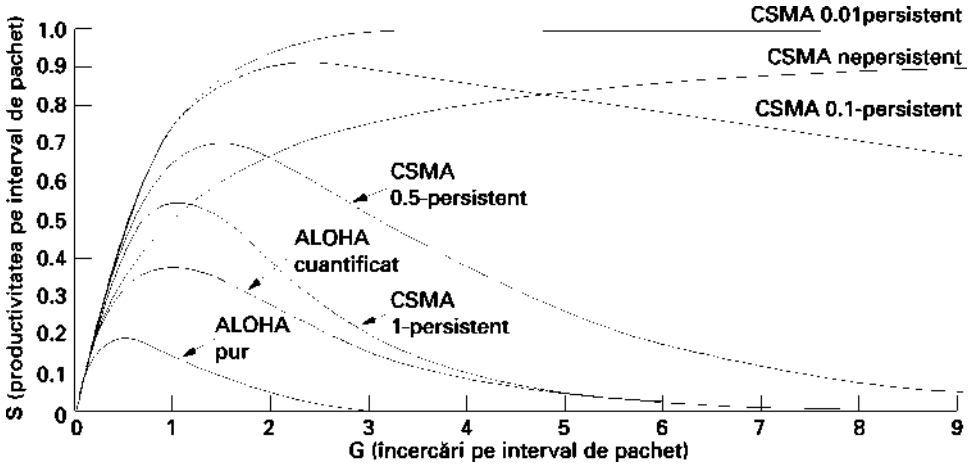


Fig. 4-4. Comparaţie între utilizările canalului în funcţie de încărcare, pentru diferite protocoale cu acces aleator.

Un al doilea protocol cu detecţie de purtătoare este **CSMA nepersistent** (**nonpersistent CSMA**). În acest protocol, o încercare conştientă de transmisie este mai puţin „lacomă" decât în cel anterior. Înainte de a emite, staţia ascultă canalul. Dacă nimeni nu emite, începe ea să emită. Dacă însă canalul este ocupat, staţia nu rămâne în continuu în ascultare, pentru a-l ocupa imediat după detectarea sfârşitului transmisiei precedente. În schimb, aşteaptă o perioadă aleatorie de timp şi apoi repetă algoritmul. Intuitiv, acest algoritm ar trebui să conducă la o utilizare mai bună a canalului, dar şi la întârzieri mai mari decât la CSMA l-persistent.

Ultimul protocol este **CSMA p-persistent (p-persistent CSMA)**. El se aplică la canalele cuantificate şi funcţionează după cum urmează. Când o staţie este gata să emită, ea ascultă canalul. Dacă acesta este liber, staţia va transmite cu o probabilitate *p.* Cu probabilitatea $q = 1 - p$, staţia va aştepta următoarea cuantă. Dacă această cuantă este de asemenea liberă, va transmite sau va aştepta din nou, cu probabilităţile p şi respectiv q. Acest proces este repetat până când cadrul este transmis sau până când o altă staţie începe să transmită. În ultimul caz, staţia se comportă ca şi când s-ar fi produs o coliziune (adică aşteaptă o perioadă aleatorie de timp şi porneşte iar). Dacă iniţial staţia detectează canalul ocupat, aşteaptă cuanta următoare şi aplică algoritmul de mai sus. Fig. 4-4

ză canalul ocupat, aşteaptă cuanta următoare şi aplică algoritmul de mai sus. Fig. 4-4 arată productivitatea în funcţie de traficul oferit pentru toate cele trei protocoale, precum şi pentru ALOHA pur şi ALOHA cuantificat.

CSMA cu detecţia coliziunii

Protocoalele CSMA persistent şi nepersistent reprezintă în mod cert o îmbunătăţire faţă de ALOHA, pentru că au grijă ca nici o staţie să nu înceapă să transmită atunci când canalul este ocupat. O altă îmbunătăţire este abandonarea transmisiei îndată ce se detectează o coliziune. Cu alte cuvinte, dacă două staţii găsesc canalul liber şi încep să transmită simultan, amândouă vor detecta coliziunea aproape imediat. Decât să îşi termine de transmis cadrele, care oricum sunt iremediabil denaturate, staţiile îşi vor termina brusc transmisia imediat după detectarea coliziunii. Terminând repede cu cadrele distruse, se salvează timp şi lărgime de bandă. Acest protocol, cunoscut sub numele de **CSMA/CD** (**Carrier Sense Multiple Access with Collision Detection**, rom: acces multiplu cu detecţia purtătoarei şi a coliziunii), este des întrebuinţat în LAN-uri în subnivelul MAC. În particular, este baza popularului Ethernet LAN, astfel încât merită efortul să îl analizăm în detaliu.

CSMA/CD, ca şi multe alte protocoale de LAN, utilizează modelul conceptual din fig. 4-5. În momentul marcat cu t_0, o staţie oarecare îşi termină de transmis cadrul. Acum orice altă staţie care are de transmis un cadru poate încerca să transmită. Dacă două sau mai multe staţii se decid să transmită simultan, se va produce o coliziune. Coliziunile pot fi detectate urmărind puterea sau lăţimea impulsului semnalului recepţionat şi comparându-le cu semnalul transmis.

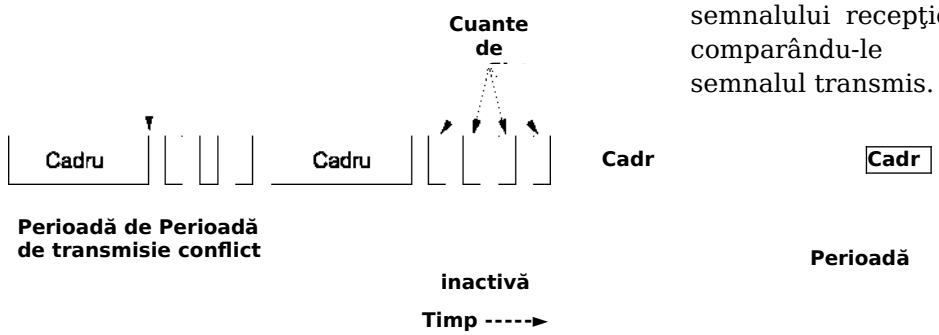


Fig. 4-5. CSMA/CD se poate afla într-una din următoarele stări: conflict, transmisie sau inactiv.

După ce o staţie a detectat o coliziune, îşi abandonează transmisia, aşteaptă o perioadă de timp oarecare şi încearcă din nou, dacă nici o altă staţie nu a început să transmită între timp. De aceea, modelul nostru pentru CSMA/CD va fi alcătuit alternativ din perioade de timp cu transmisii şi perioade de timp de conflict, având şi perioade de aşteptare, când toate staţiile tac (de exemplu, din lipsă de activitate).

Să privim acum mai îndeaproape detaliile algoritmului de tratare a conflictelor. Să presupunem că două staţii încep să transmită simultan, exact la momentul $t0$. Cât timp le va lua ca să-şi dea seama că s-a produs o coliziune? Răspunsul la această întrebare este vital pentru determinarea mărimii perioadei de conflict, deci

şi a întârzierii şi a productivităţii. Timpul minim de detectare a coliziunii este chiar timpul necesar propagării semnalului de la o staţie la alta.

Bazându-ne pe acest raţionament, am putea crede că o staţie care nu detectează nici o coliziune într-o perioadă de timp egală cu timpul de propagare pe toată lungimea cablului, perioadă măsurată de la începutul transmisiei, poate fi sigură că a ocupat canalul. Prin „ocupat” înţelegem că toate celelalte staţii ştiu că ea transmite şi nu vor interfera cu ea. Această concluzie este greşită. Să considerăm cazul cel mai defavorabil, descris în următorul scenariu. Fie τ timpul de propagare a semnalului între staţiile cele mai îndepărtate. La t_0, o staţie începe să transmită. La t_0 + (τ – e), cu o clipă înainte ca semnalul să ajungă la cea mai îndepărtată staţie, aceasta începe la rândul ei să transmită. Bineînţeles, ea detectează coliziunea aproape instantaneu şi se opreşte, dar scurta rafală de zgomot produsă de coliziune nu se va întoarce la staţia de origine decât după 2τ – e. Cu alte cuvinte, în cel mai rău caz, o staţie nu poate fi sigură că a ocupat canalul decât după ce a transmis timp de 2τ fără a detecta vreo coliziune. Din acest motiv vom modela intervalul de conflict ca un sistem ALOHA cuantificat, cu dimensiunea cuantei 2τ. Pe un cablu coaxial de 1 km, τ ® 5 ps. Simplificând, vom presupune că fiecare cuantă conţine un singur bit. Bineînţeles că, odată ce canalul a fost ocupat, o staţie poate transmite cu orice rată doreşte, nu neapărat doar 1 bit la 2τ sec.

Este important să înţelegem că detecţia coliziunii este un proces *analogic.* Echipamentul staţiei trebuie să asculte cablul în timp ce transmite. Dacă ceea ce recepţionează este diferit faţă de ceea ce transmite, înseamnă că se produce o coliziune. Aşadar, codificarea semnalului trebuie să permită detectarea coliziunilor (de exemplu, o coliziune a două semnale de 0 volţi poate fi imposibil de detectat). Din acest motiv, de obicei se utilizează codificări speciale.

Este de asemenea important de observat că o staţie care transmite trebuie să monitorizeze continuu semnalul, să asculte zgomotele care pot indica o coliziune. Din acest motiv, CSMA/CD cu un singur canal este în mod inerent un sistem jumătate-duplex (half-duplex). Este imposibil ca o staţie să transmită şi să primească cadre simultan datorită faptului că logica primirii cadrelor este activă, verificând coliziunile în timpul fiecărei transmisiuni.

Pentru a evita orice neînţelegere, e bine să notăm că nici un protocol al subnivelului MAC nu garantează o livrare corectă a cadrelor. Chiar şi în absenţa coliziunilor, receptorul poate să nu fi copiat corect cadrul din diverse motive (de exemplu, lipsă de spaţiu în zona tampon, sau o întrerupere ratată).

4.2.3 Protocoale fără coliziuni

Deşi în CSMA/CD nu apar coliziuni după ce o staţie a ocupat efectiv canalul, ele mai pot apărea în perioada de conflict. Aceste coliziuni afectează negativ performanţa sistemului, mai ales atunci când cablul este lung (adică τ mare), iar cadrele scurte. Pe măsură ce reţelele bazate pe fibre optice foarte lungi şi cu lărgime mare de bandă sunt tot mai folosite, combinaţia de valori mari pentru τ şi cadre scurte va deveni o problemă din ce în ce mai serioasă. În această secţiune vom examina câteva protocoale care rezolvă conflictul pentru canal fără nici o coliziune, nici măcar în perioada de conflict.

În protocoalele ce vor fi descrise în continuare, vom presupune că există N staţii, fiecare având o adresă unică fixă, cuprinsă între 0 şi N – 1. Nu contează dacă unele staţii sunt inactive o parte din timp. Întrebarea de bază rămâne: care staţie va primi canalul după o transmisie efectuată cu succes? Vom continua să folosim modelul din fig. 4-5, cu cuantele sale discrete de conflict.

Protocolul Bit-Map (cu hartă de biţi)

În primul nostru protocol fără coliziuni, **metoda bit-map de bază** (**basic bit-map method**), fiecare perioadă de conflict va fi formată din exact N cuante. Dacă staţia 0 are de transmis un cadru, transmite un bit 1 în timpul cuantei 0. Nici o altă staţie nu are voie să transmită în timpul acestei cuante. Fără a avea vreo legătură cu ceea ce face staţia 0, staţia 1 are ocazia să transmită un 1 în timpul cuantei 1, dar doar dacă are un cadru de transmis. În general, staţia j poate anunţa că are de transmis un

cadru inserând un bit 1 în cuanta *j.* După ce au trecut toate cele *N* cuante, fiecare staţie va cunoaşte care dintre staţii doresc să transmită. În acest moment, ele încep să transmită în ordinea crescătoare a adresei de staţie (vezi fig. 4-6).

Fig. 4-6. Protocolul bit-map

Întrucât toţi sunt de acord cine urmează, nu vor exista niciodată coliziuni. După ce ultima dintre staţiile pregătite să emită şi-a transmis cadrul, eveniment pe care toate staţiile îl pot urmări uşor, va începe o altă perioadă de conflict de *N* biţi. Dacă o staţie devine gata imediat după ce a trecut cuanta care îi corespunde, înseamnă că a ratat ocazia şi va trebui să aştepte următoarea perioadă de conflict. Protocoalele de acest gen, în care intenţia de a transmite este anunţată înainte de transmisia propriu-zisă, se numesc **protocoale cu rezervare** (reservation protocols).

Să analizăm pe scurt performanţele acestui protocol. Vom conveni ca timpul să fie măsurat în unităţi de mărimea cuantelor de un bit ale perioadei de conflict, iar cadrele de date să fie formate din *d* astfel de unităţi de timp. Practic, în condiţii de trafic slab, pachetul de biţi ai perioadei de conflict va fi transmis în mod repetat, din lipsă de cadre de date.

Să privim situaţia din punctul de vedere al unei staţii cu adresă mică, de exemplu 0 sau 1. În mod obişnuit, când ea devine gata să emită, cuanta „curentă” va fi undeva în mijlocul pachetului de biţi. În medie, o staţie va trebui să aştepte $N/2$ cuante pentru ca runda curentă să se termine şi alte N cuante până la runda următoare, înainte de a putea începe transmisia.

Perspectivele staţiilor cu adrese mari sunt ceva mai luminoase. În general, ele nu vor trebui să aştepte decât o jumătate de rundă ($N/2$ cuante) înainte de a începe să transmită. Staţiile cu adrese mari trebuie rareori să aştepte următoarea rundă. Deoarece staţiile cu adrese mici au de aşteptat în medie $1.5N$ cuante, iar cele cu adrese mari $0.5N$ cuante, media pentru toate staţiile este de N cuante. Eficienţa canalului la trafic scăzut este uşor de calculat. Încărcarea suplimentară a unui cadru este de N biţi, iar cantitatea de date este de d biţi, rezultând o eficienţă de $d/(N + d)$.

În condiţii de trafic încărcat, când toate staţiile vor să emită simultan, perioada de conflict de N biţi este împărţită la N cadre, rezultând o încărcare suplimentară de doar un bit pe cadru, adică o eficienţă de $d/(d + 1)$. Întârzierea medie pentru un cadru este egală cu suma timpului de aşteptare în interiorul staţiei, plus o întârziere suplimentară de $N(d + 1)/2$, care se adaugă atunci când ajunge la începutul cozii interne a staţiei.

Numărătoarea inversă binară

O problemă a protocolului de bază bit-map este încărcarea suplimentară de 1 bit pe staţie. Putem obţine rezultate şi mai bune utilizând adresele binare ale staţiilor. O staţie care vrea să utilizeze canalul îşi difuzează adresa ca un şir de biţi, începând cu bitul cel mai semnificativ. Se presupune că toate adresele au aceeaşi lungime.

Biţii de pe aceeaşi poziţie din adresele diferitelor staţii sunt combinaţi printr-o operaţie logică OR (SAU), iar rezultatul este citit ca o singură adresă. Vom numi acest protocol **numărătoarea inversă binară** (**binary countdown**). El este utilizat în Datakit (Fraser, 1987). Se presupune implicit că întârzierile în transmisie sunt neglijabile, astfel încât toate staţiile văd biţii transmişi practic instantaneu.

Pentru a evita conflictele, trebuie aplicată o regulă de arbitrare: de îndată ce o staţie observă că unul dintre biţii superiori ai adresei sale, conţinând un 0, a fost acoperit de un 1, renunţă să mai emită. De exemplu, dacă staţiile 0010, 0100, 1001 şi 1010 încearcă să obţină canalul în acelaşi timp, în timpul primului bit staţiile transmit 0, 0, 1 şi, respectiv, 1. Aceşti biţi sunt combinaţi printr-o operaţie SAU, rezultând un 1. Staţiile 0010 şi 0100 văd acest 1 şi ştiu că o staţie cu o adresă superioară încearcă să obţină canalul, aşa că renunţă să mai emită în runda curentă. Staţiile 1001 şi 1010 continuă.

Următorul bit este 0 şi ambele staţii continuă. Următorul bit este 1, aşa că staţia 1001 va renunţa. Câştigătoare este staţia 1010, pentru că are adresa cea mai mare. După ce a câştigat licitaţia, ea poate transmite un cadru, după care începe o nouă rundă de licitaţii. Protocolul este ilustrat în fig. 4-7. Are proprietatea că staţiile cu numere mai mari au o prioritate mai înaltă decât staţiile cu numere mai mici, ceea ce poate fi şi bine şi rău, în funcţie de context.

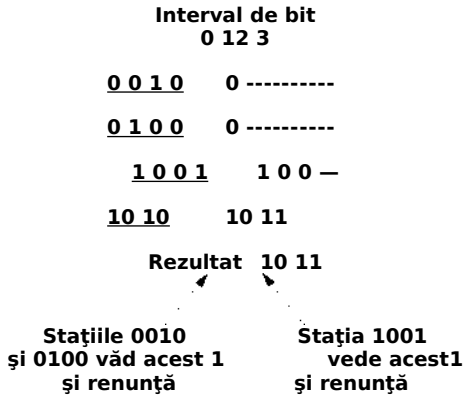


Fig. 4-7. Protocolul cu numărătoare inversă binară. O liniuţă indică tăcere.

Eficienţa canalului, în cazul acestei metode, este de $d/(d + \ln N)$. Dacă formatul cadrului a fost bine ales, astfel încât adresa emiţătorului să fie primul câmp al cadrului, chiar şi aceşti ln N biţi nu sunt pierduţi, iar eficienţa este de 100%.

Mok şi Ward (1979) au descris o variantă a numărătorii inverse binare utilizând o interfaţă paralelă în locul celei seriale. Ei au sugerat, de asemenea, utilizarea de adrese de staţie virtuale, cuprinse între 0 şi numărul staţiei câştigătoare inclusiv, adrese ce vor fi permutate după fiecare transmisie, pentru a da prioritate mai mare staţiilor care nu au mai transmis de mult. De exemplu, dacă staţiile *C, H, D, A, G, B, E* şi *F* au priorităţile 7, 6, 5, 4, 3, 2, 1 şi, respectiv, 0, atunci o transmisie cu succes a staţiei *D* o va plasa la sfârşitul listei, rezultând ordinea priorităţilor: *C, H, A, G, B, E, F, D.* Astfel, *C* rămâne virtual staţia 7, A suie de la 4 la 5, iar *D* coboară de la 5 la 0. Acum staţia *D* va putea obţine canalul numai dacă nici o altă staţie nu îl doreşte.

Numărătoarea inversă binară este un exemplu de protocol simplu, elegant şi eficient care aşteaptă să fie redescoperit. Sperăm că îşi va găsi o nouă familie cândva în viitor.

4.2.4 Protocoale cu conflict limitat

Am considerat până acum două strategii de bază pentru obţinerea canalului într-o reţea cablată: cu conflict, ca în CSMA, şi fără coliziuni. Fiecare strategie poate fi cotată după performanţe în func

ţie de doi parametri importanţi: întârzierea în condiţii de trafic scăzut şi eficienţa canalului la trafic încărcat. În condiţiile unui trafic scăzut, conflictul (adică ALOHA pur sau cuantificat) este preferat datorită întârzierilor mici. Cu cât traficul creşte, cu atât aceste metode devin tot mai puţin atractive, deoarece încărcarea suplimentară asociată cu arbitrarea canalului devine tot mai mare. Pentru protocoalele fără coliziuni apare efectul invers: la trafic scăzut, ele au întârzieri mari, dar, pe măsură ce traficul creşte, eficienţa canalului se îmbunătăţeşte în loc să se înrăutăţească, cum se întâmplă la protocoalele cu conflict.

Evident, ar fi frumos să putem combina cele mai bune proprietăţi ale protocoalelor cu conflict cu cele ale protocoalelor fără coliziuni, obţinând un nou protocol care să utilizeze varianta cu conflict la trafic scăzut, pentru a avea întârzieri mici, şi varianta fără coliziuni la trafic mare, pentru a putea oferi o eficienţă bună a canalului. Asemenea protocoale, pe care le vom numi **protocoale cu conflict limitat** (**limited contention protocols**), există şi vor încheia studiul nostru despre reţelele cu detecţie de purtătoare.

Până acum, singurele protocoale cu conflict pe care le-am studiat au fost simetrice, adică fiecare staţie încearcă să obţină canalul cu o probabilitate *p,* aceeaşi pentru toate staţiile. Un fapt destul de interesant este că performanţa globală a sistemului poate fi uneori îmbunătăţită utilizând un protocol care asociază probabilităţi diferite pentru staţii diferite.

Înainte de a trece la protocoalele asimetrice, să trecem succint în revistă performanţele cazului simetric. Să presupunem că există k staţii care concurează pentru obţinerea accesului la canal. Fiecare are o probabilitate p de a transmite în timpul fiecărei cuante. Probabilitatea ca o staţie să obţină canalul în timpul unei cuante este $kp(1-p)^{k-1}$. Pentru a obţine valoarea optimă pentru p, derivăm în raport cu p, egalăm rezultatul cu zero şi rezolvăm pentru p. Vom obţine că valoarea cea mai bună a lui p este $1/k$. Substituind $p = 1/k$, obţinem probabilitatea

$$\Pr[\text{succes cu } p \quad \left(\frac{k-1}{k}\right)^{k-1} \qquad (4-$$

Această probabilitate este reprezentată în fig. 4-8. Pentru un număr mic de staţii şansele de succes sunt mari, dar probabilitatea scade către o valoare asimptotică de *1/e* înainte chiar ca numărul staţiilor să atingă valoarea cinci.

Din fig. 4-8 reiese clar că probabilitatea ca o staţie să obţină canalul poate fi crescută doar reducând concurenţa. Protocoalele cu conflict limitat fac exact acest lucru. Mai întâi, ele împart staţiile în grupuri (nu neapărat disjuncte). Doar membrilor grupului 0 li se permite să concureze pentru cuanta 0. Dacă unul din ei reuşeşte, ocupă canalul şi îşi transmite cadrul. În cazul în care cuanta rămâne neîntrebuinţată sau apare o coliziune, membrii grupului 1 vor concura pentru cuanta 1 etc. Făcând o împărţire corectă a staţiilor în grupuri, numărul de conflicte pentru fiecare cuantă poate fi redus, aducând performanţa corespunzătoare fiecărei cuante către extrema stânga a fig. 4-8.

Trucul constă în modul în care asociem staţiile cuantelor. Înainte de a analiza

cazul general, să considerăm câteva cazuri particulare. La o extremă, fiecare grup are un singur membru. O astfel de împărţire garantează că niciodată nu vom avea coliziuni, pentru că cel mult o staţie concurează pentru o cuantă. Am văzut astfel de protocoale anterior (de exemplu, numărătoarea inversă binară). Următorul caz particular este împărţirea în grupuri de câte două staţii. Probabilitatea ca amândouă să încerce să transmită în timpul unei cuante este p^2, ceea ce pentru un p mic este o valoare neglijabilă. Pe măsură ce unei cuante îi sunt asociate mai multe staţii, probabilitatea unei coliziuni creşte, în schimb lungimea pachetului de biţi, necesar pentru a da fiecăruia o şansă, se micşorează. Cazul limită este un singur grup conţinând toate staţiile (adică ALOHA cuantificat). Ceea ce ne trebuie este o

cale de a asocia staţii cuantelor în mod dinamic, cu multe staţii pe cuantă atunci când traficul este scăzut şi puţine staţii pe cuantă (sau chiar una singură) atunci când traficul este mare.

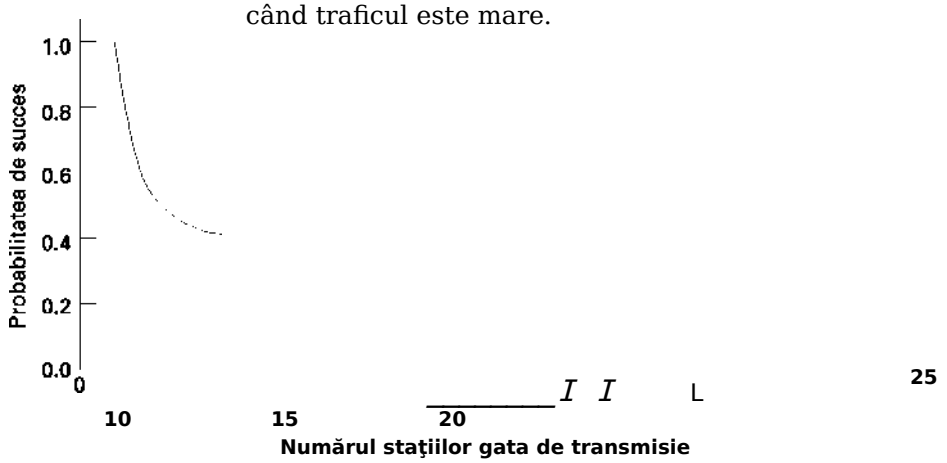


Fig. 4-8. Probabilitatea de obţinere a unui canal cu conflict simetric.

Protocolul cu parcurgere arborescentă adaptivă

O cale foarte simplă de a face o asociere bună este utilizarea algoritmului conceput de armata Statelor Unite în scopul testării pentru sifilis a soldaţilor în timpul celui de-al doilea război mondial (Dorfman, 1943). Pe scurt, armata preleva eşantioane de sânge de la N soldaţi. O porţiune din fiecare eşantion era pusă în acelaşi tub de test. Acest eşantion mixat era apoi testat pentru anticorpi. Dacă nu era găsit nici un anticorp, toţi soldaţii din grup erau declaraţi sănătoşi. Dacă însă

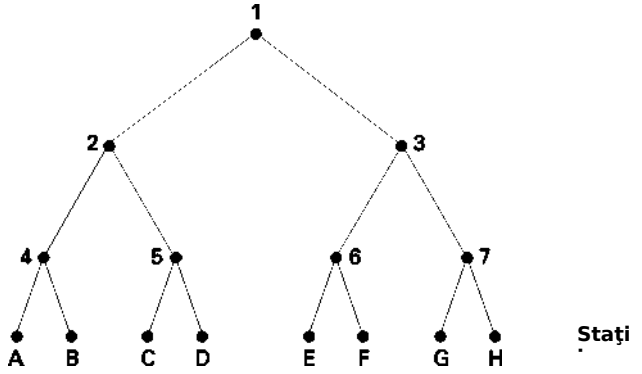


Fig. 4-9. Arborele pentru opt

erau prezenţi anticorpi, erau preparate două noi eşantioane mixte, unul corespunzător soldaţilor de la 1 la $N/2$, iar altul corespunzător celorlalţi. Procesul era repetat recursiv până când erau determinaţi soldaţii infectaţi.

Pentru versiunea informatică a acestui algoritm (Capetanakis, 1979), cel mai simplu este să ne închipuim staţiile ca fiind frunzele unui arbore binar, ca în fig. 4-9. În prima cuantă de conflict care urmează după un cadru transmis cu succes, şi anume cuanta 0, toate staţiile au permisiunea de a încerca ocuparea canalului. Dacă numai una din ele încearcă, foarte bine. Dacă s-a produs o coliziune, atunci, în timpul cuantei 1, doar staţiile de sub nodul 2 din arbore pot concura. Dacă una din ele obţine canalul, cuanta care urmează cadrului ce va fi transmis este rezervat

pentru staţiile de sub nodul 3. Dacă, pe de altă parte, două sau mai multe staţii de sub nodul 2 vor să transmită, se va produce o coliziune în timpul cuantei 1, caz în care va fi rândul nodului 4 în timpul cuantei 2.

În principiu, dacă apare o coliziune în timpul cuantei 0, este cercetat întregul arbore în adâncime, pentru a localiza toate staţiile gata să transmită. Fiecare cuantă de un bit este asociată unui nod particular din arbore. Dacă se produce o coliziune, căutarea este continuată recursiv cu fiii stâng şi drept ai nodului. Dacă o cuantă de un bit este liber sau dacă o singură staţie transmite în timpul ei, căutarea nodului pentru această cuantă se poate opri, pentru că toate staţiile gata să transmită au fost localizate (dacă erau mai multe decât una s-ar fi produs o coliziune).

Atunci când încărcarea sistemului este mare, nu prea merită să dedicăm cuanta 0 nodului 1, pentru că acest lucru ar avea sens doar în eventualitatea – destul de puţin probabilă – ca o singură staţie să aibă un cadru de transmis. Similar, se poate argumenta că nodurile 2 şi 3 pot fi lăsate la o parte din aceleaşi motive. În termeni mai generali, întrebarea este: la ce nivel din arbore ar trebui să înceapă căutarea? Desigur, cu cât traficul este mai mare, cu atât căutarea trebuie să înceapă mai de jos. Vom considera că fiecare staţie deţine o estimare corectă a numărului de staţii gata să transmită, de exemplu q, obţinută din monitorizarea traficului recent.

Pentru început, să numărăm nivelurile arborelui începând de la vârf, cu nodul 1 din fig. 4-9 pe nivelul 0, nodurile 2 şi 3 pe nivelul 1 etc. Observaţi că fiecare nod de pe nivelul i are dedesubt o fracţiune de 2^{-i} din totalul staţiilor. Dacă cele q staţii gata să transmită sunt uniform distribuite, numărul celor care se află sub un anumit nod de pe nivelul i este $2^{-i}q$. Intuitiv, ar trebui ca nivelul optim de începere a căutării să fie cel pentru care numărul mediu de staţii care vor să transmită în timpul unei cuante este 1, adică nivelul la care $2^{-i}q = 1$. Rezolvând această ecuaţie vom găsi că $i = \log_2 q$.

Au fost descoperite numeroase îmbunătăţiri ale algoritmului de bază, care sunt discutate în detaliu de Bertsekas şi Gallager (1992). De exemplu, să considerăm cazul în care staţiile G şi H vor să transmită. La nodul 1 se va produce o coliziune, aşa că va fi încercat 2, care va fi găsit liber. Este fără sens să încercăm nodul 3 pentru că este sigur că vom avea o coliziune (ştim că două sau mai multe staţii de sub 1 vor să transmită şi nici una dintre ele nu se află sub 2, deci toate sunt sub 3). Încercarea lui 3 poate fi sărită şi se trece la 6. Dacă nici această încercare nu dă nici un rezultat, 7 poate fi sărit şi este încercat G în continuare.

4.2.5 Protocoale cu acces multiplu cu divizarea frecvenţei

O abordare diferită a problemei alocării canalului o reprezintă împărţirea acestuia în subcanale utilizând FDM, TDM, sau amândouă, şi alocarea lor dinamică după necesităţi. Astfel de metode sunt frecvent utilizate în LAN-urile cu fibră optică pentru a permite ca transmisiuni diferite să utilizeze lungimi de undă (adică frecvenţe) diferite în acelaşi timp. În această secţiune vom examina un astfel de protocol (Humblet ş.a., 1992).

O cale simplă de construire a unui LAN cu fibră optică este utilizarea unui cuplor pasiv de tip stea (vezi fig. 2-10). Două fibre de la fiecare staţie intră într-un cilindru

de sticlă. O fibră este pentru transmisia către cilindru iar cealaltă pentru transmisia de la cilindru. Emisia de lumină de la oricare din staţii iluminează cilindrul şi poate fi detectată de toate celelalte staţii. Stelele pasive pot cupla până la sute de staţii.

Pentru a permite transmisiuni multiple simultane, spectrul este divizat în canale (benzi de frecvenţă), ca în fig. 2-24. În acest protocol, **WDMA** (**Wavelength Division Multiple Access**, rom: acces multiplu cu divizarea frecvenţei), fiecărei staţii îi sunt asociate două canale. Un canal îngust este folosit drept canal de control pentru semnalizarea către staţie, iar unul larg pentru ca staţia să poată trimite cadre de date prin el.

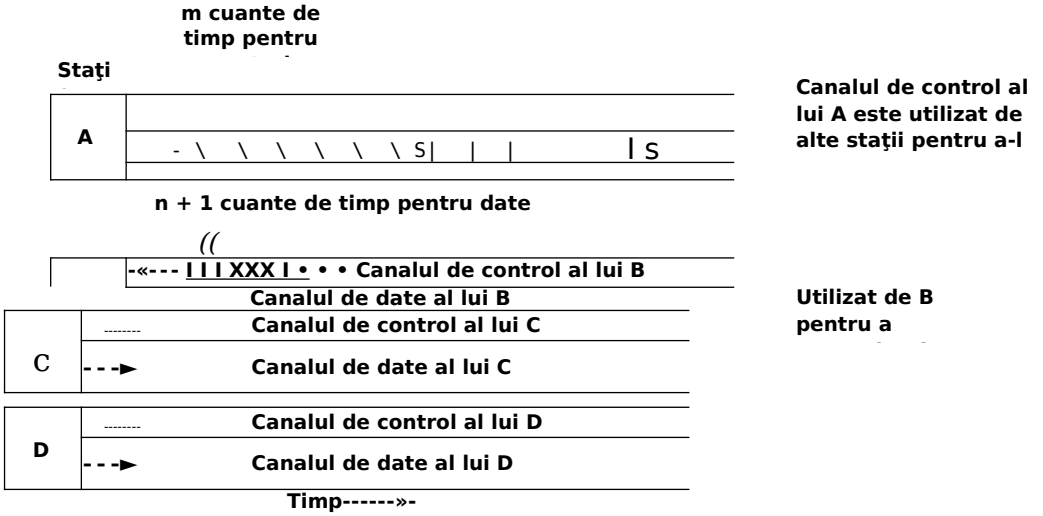

m cuante de timp pentru
Staţi
A
Canalul de control al lui A este utilizat de alte staţii pentru a-l
n + 1 cuante de timp pentru date
Canalul de control al lui B
Canalul de date al lui B
Utilizat de B pentru a
C
Canalul de control al lui C
Canalul de date al lui C
D
Canalul de control al lui D
Canalul de date al lui D
Timp------»-

Fig. 4-10. Acces multiplu cu divizarea frecvenţei.

Fiecare canal este împărţit în cuante de timp, ca în fig. 4-10. Fie *m* numărul de cuante ale canalului de control şi *n* + 1 numărul de cuante ale canalului de date, dintre care *n* sunt pentru date şi ultima este utilizată de staţie pentru a-şi raporta starea (în principal, care dintre cuantele ambelor canale sunt libere). Pe ambele canale secvenţa de cuante se repetă la infinit, cu cuanta 0 marcată special pentru ca cei care iau parte mai târziu la transmisie să o poată detecta. Toate canalele sunt sincronizate de un unic ceas global.

Protocolul suportă trei clase de trafic: (1) trafic orientat pe conexiune, cu rată constantă de date (cum este semnalul video necomprimat), (2) trafic orientat pe conexiune, cu rată variabilă de date (cum este transferul de fişiere) şi (3) trafic de datagrame, cum sunt pachetele UDP. Pentru cele două protocoale orientate pe conexiune, ideea este că dacă *A* vrea să comunice cu *B,* trebuie să insereze mai întâi un cadru CONNECTION REQUEST (cerere conectare) într-o cuantă liberă de pe canalul de control al lui *B*. Dacă *B* acceptă, comunicaţia se poate desfăşura pe canalul de date al lui *A*.

Fiecare staţie are doi emiţători şi doi receptori, după cum urmează:

1. Un receptor cu lungime de undă fixă pentru ascultarea propriului canal de control.
2. Un emiţător reglabil pentru comunicarea pe canalul de control al altei staţii.
3. Un emiţător cu lungime de undă fixă pentru emisia cadrelor de date.
4. Un receptor reglabil pentru selectarea emiţătorului de ascultat.

Cu alte cuvinte, fiecare staţie îşi ascultă propriul canal de control pentru cererile care sosesc, dar trebuie să se regleze pe frecvenţa emiţătorului pentru a primi datele. Reglarea frecvenţei este realizată cu un interferometru Fabry-Perot sau Mach-Zehnder, care elimină prin filtrare toate frecvenţele, cu excepţia celei dorite.

Să urmărim acum modul în care staţia *A* stabileşte un canal de comunicaţie de clasă 2 cu staţia *B* pentru, să zicem, un transfer de fişiere. Mai întâi, *A* îşi reglează receptorul de date pe frecvenţa canalului de date al lui *B* şi aşteaptă cuanta de

stare. Această cuantă precizează care cuante de control sunt ocupate şi care sunt libere. De exemplu, în fig. 4-10 se observă că din cele opt cuante de control ale lui *B,* 0, 4 şi 5 sunt libere. Restul sunt ocupate (fapt indicat prin cruciuliţe).

A îşi alege una din cele trei cuante de control, să zicem 4, şi îşi inserează mesajul CONNECT REQUEST în ea. Cum *B* îşi ascultă permanent canalul de control, vede cererea şi o aprobă acordând cuanta 4 lui *A*. Această decizie este anunţată în cuanta de stare a canalului de control. Atunci când *A* vede anunţul, va şti că s-a stabilit o conexiune unidirecţională. Dacă *A* cerea o conexiune bidirecţională, *B* ar fi trebuit să repete acelaşi algoritm cu *A*.

Este posibil ca în timp ce *A* căuta să ocupe cuanta 4 de control a lui *B*, *C* să facă acelaşi lucru. Nici o staţie nu o va obţine şi amândouă vor observa eşecul urmărind cuanta de stare din canalul de date al lui *B*. Ele vor aştepta în continuare un interval de timp aleatoriu, după care vor încerca din nou.

În acest moment, fiecare staţie are o cale fără conflicte pentru trimiterea de scurte mesaje de control către cealaltă. Pentru a realiza transferul de fişiere, *A* va trimite către *B* un mesaj de control, spunând, de exemplu, „Te rog uită-te la următoarea cuantă 3 cu date de ieşire de la mine. În ea se află un cadru de date pentru tine”. Când *B* primeşte mesajul de control, îşi va regla receptorul pe canalul de ieşire al lui *A* pentru a citi cadrul de date. Bazându-se pe un protocol de nivel mai înalt, *B* poate utiliza acelaşi mecanism pentru a trimite înapoi o confirmare, dacă doreşte.

De notat că apare o problemă când *A* şi *C* au conexiuni către *B* şi fiecare îi spune să se uite la cuanta 3. *B* va alege una dintre ele la întâmplare, iar cealaltă transmisie va fi pierdută.

La trafic constant este utilizată o variantă a acestui protocol. Atunci când *A* cere o conexiune, ea spune în acelaşi timp ceva de genul: este în regulă dacă îţi voi trimite câte un cadru în fiecare cuantă 3? Dacă *B* poate accepta (adică nu şi-a luat nici un angajament pentru cuanta 3), este stabilită o conexiune cu lărgime de bandă garantată. Dacă nu, *A* poate încerca din nou cu o altă propunere, în funcţie de cuantele de ieşire libere.

Traficul de clasă 3 (datagrame) utilizează o altă variantă. În loc să scrie un mesaj CONNECTION REQUEST în cuanta de control pe care tocmai a găsit-o (4), va scrie un mesaj DATA FOR YOU ÎN SLOT 3 (în cuanta 3 se află date pentru tine). Dacă *B* este liberă în timpul următoarei cuante 3 de date, transmisiunea va reuşi. Altfel, cadrul de date se va pierde. În acest fel nu vom avea niciodată nevoie de conexiuni.

Sunt posibile mai multe variante ale întregului protocol. De exemplu, în loc să îi asigurăm fiecărei staţii propriul canal de control, toate staţiile pot partaja un singur canal de control. Fiecărei staţii îi este asociat un bloc de cuante în fiecare grup, multiplexând astfel mai multe canale virtuale într-un singur canal fizic.

De asemenea ne putem descurca cu un singur emiţător reglabil şi un singur receptor reglabil pe staţie, divizând canalul fiecărei staţii în m cuante de control, urmate de $n + 1$ cuante de date. Dezavantajul constă în faptul că emiţătorii trebuie să aştepte mai mult pentru a obţine o cuantă de control, iar cadrele de date consecutive vor fi separate din cauza informaţiilor de control de pe canal.

Au fost propuse numeroase alte protocoale WDMA, care se deosebesc prin detalii. Unele au un singur canal de control, altele au mai multe. Unele iau în

considerare întârzierea de propagare, altele nu; unele consideră timpul de reglare a frecvenţei ca făcând parte explicit din model, altele îl ignoră. De asemenea protocoalele se deosebesc prin complexitatea prelucrării, productivitate şi scalabilitate. Când sunt folosite un număr ridicat de frecvenţe sistemul poate fi numit **DWDM (Dense Wavelength Division Multiplexing –** acces multiplu dens cu divizarea frecvenţei). Pentru mai multe informaţii, vezi (Bogineni ş.a., 1993; Chen, 1994; Goralski, 2001; Kartopoulos, 1999; Levine şi Akyildiz, 1995).

4.2.6 Protocoale pentru reţele LAN fără fir

Pe măsură ce numărul de echipamente de calcul şi comunicaţie creşte, acelaşi lucru se întâmplă şi cu nevoia lor de conectare la lumea exterioară. Chiar şi primele telefoane portabile aveau posibilitatea de a se conecta la alte telefoane. Primele calculatoare portabile nu au avut această posibilitate, dar curând după aceea, modemurile au devenit un lucru obişnuit. Pentru a comunica, aceste calculatoare trebuiau să fie conectate la o priză telefonică de perete. Necesitatea unei conexiuni prin cablu la o reţea fixă însemna că de fapt calculatoarele, deşi erau portabile, nu erau mobile.

Pentru a obţine o adevărată mobilitate, calculatoarele portabile trebuie să utilizeze pentru comunicaţie semnale radio (sau infraroşii). Astfel, utilizatorii dedicaţi pot citi sau trimite poşta electronică în timp ce merg cu maşina sau cu vaporul. Un sistem de calculatoare portabile care comunică prin radio poate fi privit ca un LAN fără fir. Aceste LAN-uri au proprietăţi oarecum diferite faţă de LAN-urile convenţionale şi necesită protocoale speciale pentru subnivelul MAC. În această secţiune vom examina câteva din aceste protocoale. Mai multe informaţii despre reţelele locale fără fir pot fi găsite în (Geier, 2002; O'Hara şi Petrick, 1999).

O configuraţie obişnuită pentru un LAN fără fir este o clădire cu birouri, cu staţiile de bază amplasate strategic în jurul clădirii. Toate staţiile de bază sunt interconectate prin cabluri de cupru sau fibră optică. Dacă puterea de emisie a staţiilor de bază şi a calculatoarelor portabile este reglată la o rază de acţiune de 3 sau 4 metri, atunci fiecare cameră devine o singură celulă, iar întreaga clădire devine un mare sistem celular, ca în sistemele de telefonie celulară tradiţională, pe care le-am studiat în Cap. 2. Însă, spre deosebire de sistemele de telefonie celulară, fiecare celulă are un singur canal, acoperind întreaga lărgime de bandă disponibilă şi acoperind toate staţiile din respectiva celulă. În mod normal, lărgimea de bandă a canalului este de 11-54 Mbps.

În discuţia care urmează vom presupune, pentru simplificare, că toţi emiţătorii radio au un domeniu fix. Atunci când un receptor se află în raza a doi emiţători activi, semnalul rezultat va fi, în general, amestecat şi neutilizabil (cu câteva excepţii care vor fi discutate mai târziu). E important să ne dăm seama că în unele LAN-uri fără fir nu toate staţiile se află în acelaşi domeniu, ceea ce duce la o serie de complicaţii. Mai mult, pentru LAN-uri de incintă fără fir, prezenţa pereţilor între staţii poate avea un impact major asupra domeniului efectiv al fiecărei staţii.

O abordare naivă în construirea unui LAN fără fir o constituie încercarea de utilizare a CSMA, prin ascultarea celorlalte transmisiuni şi transmisia numai în cazul în care nimeni nu transmite. Problema este că acest protocol nu este chiar potrivit, pentru că ceea ce contează este interferenţa la receptor, nu la emiţător. Pentru a vedea natura problemei, să privim fig. 4-11, în care apar patru staţii nelegate prin cablu. Pentru ceea ce vrem să arătăm nu contează care sunt staţii de bază şi care sunt calculatoare portabile. Domeniul (de recepţie) radio are proprietatea că *A* şi *B* sunt fiecare în domeniul celeilalte şi pot interfera una cu cealaltă. Şi *C* poate să interfereze atât cu *B* cât şi cu *D,* dar nu cu *A*.

Fig. 4-11. Un LAN fără fir. (a) A transmite. (b) B transmite.

Să considerăm mai întâi ce se întâmplă atunci când *A* transmite către *B,* ca în fig. 4-11 (a). Dacă *C* ascultă mediul, ea nu o va auzi pe *A* pentru că *A* este în afara domeniului ei, trăgând concluzia falsă că poate transmite. Dacă *C* începe să transmită, ea va interfera la *B* cu cadrul de la *A*, distrugându-l. Problema staţiei care nu poate detecta un potenţial competitor la mediu pentru că se află prea departe este numită uneori **problema staţiei ascunse** (**hidden station problem**).

Să considerăm acum situaţia inversă: *B* transmite către *A*, ca în fig. 4-11(b). Dacă *C* ascultă mediul, va sesiza transmisia şi va deduce în mod incorect că nu poate transmite către *D,* când de fapt o asemenea transmisie ar cauza o proastă recepţie doar în zona cuprinsă între *B* şi *C*, unde nu se află nici unul dintre receptorii vizaţi. Această situaţie se mai numeşte şi **problema staţiei expuse** (**exposed station problem**).

Problema este că înainte de a începe o transmisiune, o staţie doreşte să ştie dacă în preajma receptorului se desfăşoară sau nu vreo activitate. CSMA sesizează acest lucru prin simpla detecţie a purtătoarei. Prin cablu, toate semnalele se propagă la toate staţiile, aşa că, la un moment dat, poate avea loc o singură transmisie, indiferent de zona sistemului în care se desfăşoară ea. Într-un sistem bazat pe unde radio cu domeniu mic, se pot desfăşura mai multe transmisiuni simultan, dacă acestea au destinaţii diferite şi aceste destinaţii au domenii disjuncte.

Altă cale de abordare a acestei probleme este să ne închipuim o clădire de birouri în care fiecare angajat are un calculator portabil nelegat prin cablu. Să presupunem că Linda vrea să îi transmită un mesaj lui Milton. Calculatorul Lindei ascultă mediul local şi, nedetectând nici o activitate, începe să transmită. Totuşi, se mai poate produce o coliziune în biroul lui Milton, pentru că o a treia persoană îi transmitea deja dintr-un alt loc, atât de departe de Linda, încât calculatorul ei nu a putut detecta acest lucru.

MACA şi MACAW

Unul dintre primele protocoale concepute pentru LAN-uri fără fir este **MACA** (**Multiple Access with Collision Avoidance –** acces multiplu cu evitarea coliziunii) (Karn, 1990). El a fost utilizat ca bază pentru standardul de LAN fără fir IEEE 802.11. Ideea de bază care stă în spatele său este ca emiţătorul să stimuleze receptorul să emită un scurt cadru, astfel încât staţiile apropiate să poată detecta această transmisiune şi să nu emită şi ele pe durata cadrului (mare) de date care urmează. MACA este ilustrat în fig. 4-12.

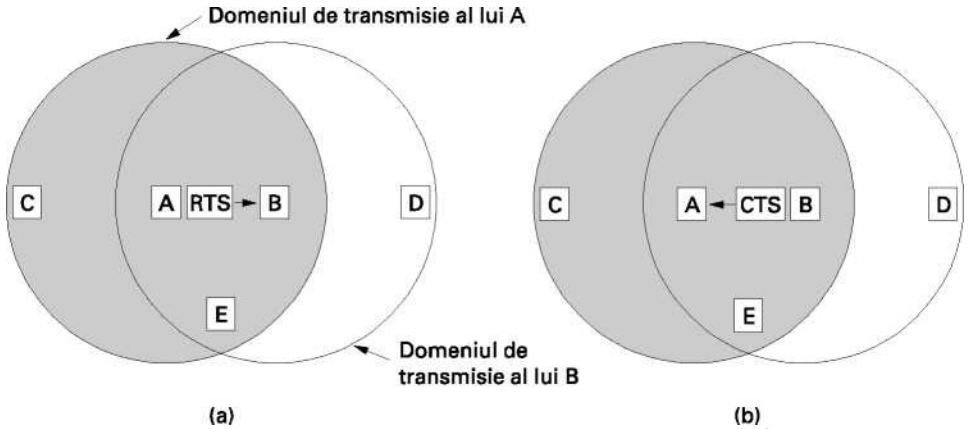


Fig. 4-12. Protocolul MACA. (a) *A* emite un RTS către B. (b) *B* îi răspunde lui A cu un CTS.

Să vedem acum modul în care *A* îi trimite un cadru lui *B. A* începe prin a emite un cadru RTS (Request To Send, rom: cerere de emisie) către *B*, ca în fig. 4-12(a). Acest scurt cadru (30 de octeţi) conţine lungimea cadrului de date care va urma. Apoi *B* răspunde cu un cadru CTS (Clear To Send, rom: aprobare transmisie), ca în fig. 4-12(b). Cadrul CTS conţine lungimea datelor (copiată din cadrul RTS). La recepţia cadrului CTS, *A* începe transmisia.

Să urmărim acum modul în care reacţionează staţiile care recepţionează vreunul din aceste cadre. Orice staţie care aude RTS se află în mod cert în apropierea lui *A* şi trebuie să tacă suficient de mult timp pentru ca să poată fi trimis un CTS înapoi la *A*, fără conflicte. Orice staţie care recepţionează CTS se află în mod cert în apropiere de *B* şi trebuie să tacă în timpul transmisiei de date în curs, a cărei lungime o poate afla examinând cadrul CTS.

În fig. 4-12, *C* se află în domeniul lui *A*, însă nu în domeniul lui *B*. De aceea va auzi RTS de la *A*, dar nu şi CTS de la *B*. Cât timp nu interferează cu CTS, ea este liberă să transmită în timp ce cadrul de date este emis. În schimb *D* este în domeniul lui *B*, dar nu şi în cel al lui *A*. Nu aude RTS, dar aude CTS. Recepţionând CTS, îşi va da seama că este aproape de o staţie care este pe cale să primească un cadru, aşa că se va abţine de la a emite ceva până când, după calculele sale, acel cadru se va termina. Staţia *E* aude ambele mesaje de control şi, ca şi *D*, trebuie să tacă până la terminarea cadrului de date.

În ciuda acestor precauţii, încă mai pot apărea coliziuni. De exemplu, *B* şi *C* ar putea transmite simultan cadre RTS către *A*. Ele vor intra în coliziune şi se vor pierde. În eventualitatea unei coliziuni, un emiţător care nu a avut succes (adică unul care nu aude un CTS în intervalul de timp prevăzut) va aştepta o perioadă de timp aleatorie şi va încerca din nou. Algoritmul utilizat este cel de re- gresie exponenţială binară, pe care îl vom studia când vom ajunge la LAN-ul IEEE 802.3.

Bazat pe studii de simulare a MACA, Bharghavan ş.a. (1994) au reuşit până la urmă să îmbunătăţească performanţele MACA şi au redenumit noul lor protocol **MACAW**. La început, ei au observat că, fără confirmări ale nivelului legătură de date, cadrele pierdute nu erau retransmise până când, mult mai târziu, nivelul transport le observa absenţa. Au rezolvat această problemă introducând un cadru

de confirmare ACK după fiecare cadru de date transmis cu succes. Tot ei au mai observat că CSMA are o oarecare utilitate, şi anume să oprească o staţie de la a transmite un RTS concomitent cu o altă staţie apropiată care face acelaşi lucru către aceeaşi destinaţie, aşa că a fost adăugată şi detecţia de purtătoare. În plus, ei au mai decis să execute algoritmul de regresie separat pentru fiecare flux de date (pereche sursă-destinaţie), iar nu pentru fiecare staţie. Această schimbare îmbunătăţeşte echitatea protocolului. În final, pentru a îmbunătăţi performanţele sistemului, s-au mai adăugat: un mecanism ce permite staţiilor să schimbe informaţii despre congestia reţelei şi o cale de a face ca algoritmul de regresie să reacţioneze mai puţin violent la problemele temporare.

4.3 ETHERNET

Am terminat acum discuţia noastră generală despre protocoalele de alocare a canalelor în teorie, deci este timpul să vedem cum se aplică aceste principii sistemelor reale – în particular, LAN-urilor. După cum am discutat în secţiunea 1.5.3, IEEE a standardizat un număr de reţele locale şi metropolitane sub numele de IEEE 802. Câteva au supravieţuit, dar nu multe, după cum am văzut în fig. 1 38. Unii dintre cei care cred în reîncarnare se gândesc că Charles Darwin s-a întors ca membru al IEEE Standards Association pentru a elimina reţelele neadaptate. Cei mai importanţi dintre supravieţuitori sunt: 802.3 (Ethernet) şi 802.11 (LAN fără fir). În ceea ce priveşte 802.15 (Bluetooth) şi 802.16 (MAN fără fir), este prea devreme pentru a ne pronunţa. Vă sfătuim să consultaţi ediţia a 5-a a acestei cărţi ca să aflaţi. Atât 802.3 şi 802.11 au niveluri fizice diferite şi subniveluri MAC diferite, dar ele converg asupra aceluiaşi subnivel logic de control al conexiunii (LLC) (definit în 802.2), astfel încât au aceeaşi interfaţă cu nivelul de reţea.

Am introdus Ethernetul în Sec.1.5.3 şi nu vom mai repeta aici aceleaşi informaţii. În continuare ne vom concentra asupra detaliilor tehnice ale Ethernetului, protocoalele şi realizările recente în Ethernet-ul de mare viteză (gigabit). Din moment ce Ethernet şi IEEE 802.3 sunt aproape identice, cu excepţia a două detalii minore pe care le vom discuta în curând, mulţi oameni folosesc termenii „Ethernet” şi „IEEE 802.3” ca sinonime, astfel încât şi noi vom face acelaşi lucru. Pentru mai multe informaţii despre Ethernet, vezi (Bradley şi Riley, 1999; Seifert, 1998; Spurgeon, 2000).

4.3.1 Cablarea Ethernet

Întrucât numele „Ethernet” se referă la cablu (eterul), să pornim discuţia noastră de aici. În mod obişnuit, sunt utilizate patru tipuri de cabluri, după cum se arată în fig. 4-13.

Nume	Cablu	Seg. maxim	Noduri / seg.	Avantaje
10Base5	coaxial gros	500 m	100	Cablul original, în prezent ieşit din uz

10Base2	coaxial subţire	185 m	30	Nu este nevoie de hub
10Base-T	perechi torsadate	100 m	1024	Cel mai ieftin sistem
10Base-F	Fibră optică	2000 m	1024	Cel mai bun între clădiri

Fig. 4-13. Cele mai obişnuite tipuri de cablare Ethernet.

Din punct de vedere istoric, cablul **10Base5**, numit popular şi **Ethernet gros** (**thick Ethernet**), a fost primul. El se aseamănă cu un furtun galben de grădină cu semne la fiecare 2.5 metri pentru a arăta unde vin conectorii (Standardul 802.3 nu *impune* de fapt cabluri de culoare galbenă, dar *sugerează* acest lucru). Conexiunile cu el sunt făcute în general utilizând **conectori-vampir** (**vampire taps**), la care un pin este introdus cu *mare* grijă până în miezul cablului coaxial. Notaţia 10Base5 înseamnă că funcţionează la 10 Mbps, utilizează semnalizare în banda de bază şi poate suporta segmente de până la 500 metri. Primul număr reprezintă viteza în Mbps. Apoi urmează cuvântul „Base” (uneori „BASE”) pentru a indica transmisia în banda de bază. Exista mai demult o variantă în banda largă, 10Broad36, dar nu s-a impus pe piaţă şi a dispărut. În fine, dacă mediul de transmisie este cablul coaxial, lungimea sa apare rotunjită în unităţi de 100m după „Base”.

Istoric vorbind, al doilea tip de cablu a fost **10Base2**, sau **Ethernet subţire** (**thin Ethernet**), care, spre deosebire de Ethernet gros „ca un furtun de grădină”, se îndoaie uşor. Conexiunile cu el sunt făcute utilizând conectori standard industriali BNC pentru a forma joncţiuni în T, mai curând decât conectori-vampir. Aceştia sunt mai uşor de folosit şi mai siguri. Ethernetul subţire este mult mai ieftin şi mai uşor de instalat, dar el poate suporta lungimi ale cablului de maxim 185 de metri pe segment, fiecare segment putând trata numai 30 de calculatoare.

Detectarea întreruperilor de cablu, a conectorilor proşti sau a conectorilor desprinşi poate fi o problemă majoră pentru ambele medii de transmisie. Din acest motiv au fost dezvoltate tehnici care să le detecteze. În esenţă, în cablu este injectat un impuls cu o formă cunoscută. Dacă impulsul întâlneşte un obstacol sau ajunge la capătul cablului, va fi generat un ecou care este trimis înapoi. Măsurând cu grijă timpul scurs între emiterea impulsului şi recepţionarea ecoului, este posibilă localizarea originii ecoului. Această tehnică este numită **reflectometrie în domeniul timp** (**time domain reflectometry**).

Problemele asociate cu găsirea întreruperilor de cablu au condus sistemele către un alt tip de model de cablare, în care toate staţiile au un cablu care duce la un **concentrator** (**hub**). De obicei, aceste fire sunt perechi torsadate ale companiei de telefoane, deoarece majoritatea clădirilor cu birouri sunt deja cablate în acest fel şi, în mod normal, există o mulţime de perechi disponibile. Această strategie se numeşte **10Base-T**. Concentratorii nu pot ţine într-o memorie tampon traficul pe care îl transferă. Vom discuta mai târziu în acest capitol o versiune îmbunătăţită a acestei idei (comutatoarele), care au mecanisme de păstrare a traficului primit într-o memorie tampon.

Aceste trei strategii de cablare sunt ilustrate în fig. 4-14. Pentru 10Base5, în jurul cablului este prins strâns un **transiver** (**transceiver**), astfel încât conectorul său face contact cu miezul cablului. Transiverul conţine partea de electronică care se ocupă cu detecţia purtătoarei şi cu detecţia coliziunilor. Atunci când este detectată o coliziune, transiverul trimite pe cablu un semnal nepermis special,

pentru a se asigura că şi celelalte transivere îşi dau seama că s-a produs o coliziune.

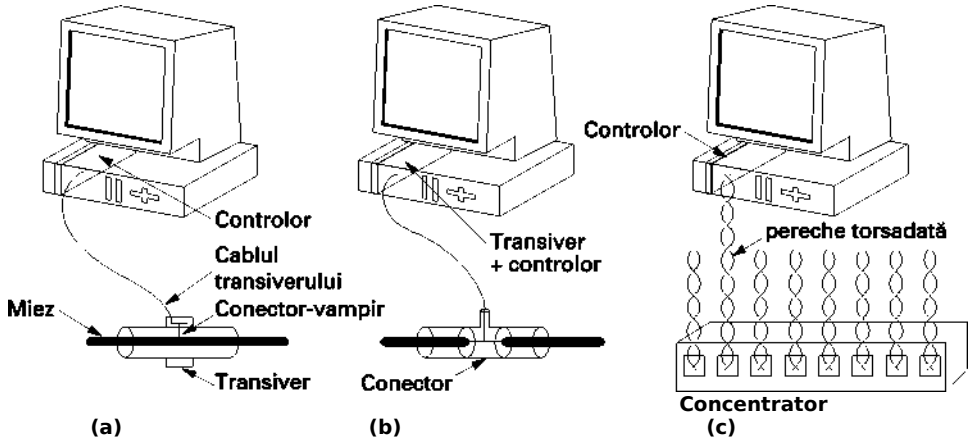


Fig. 4-14. Trei tipuri de cablare 802.3. (a) 10Base5. (b) 10Base2. (c)10Base-T.

La 10Base5, un **cablu de transiver** (**transceiver cable**) conectează transiverul cu o placă de interfaţă din calculator. Cablul transiverului poate avea până la 50 de metri lungime şi conţine cinci perechi torsadate izolate individual. Două dintre perechi sunt pentru datele de intrare şi respectiv datele de ieşire. Alte două sunt pentru semnalele de control de intrare şi de ieşire. A cincea pereche, care nu este întotdeauna folosită, permite calculatorului să alimenteze electronica transiverului. Pentru a reduce numărul de transivere necesare, unele transivere permit să le fie ataşate până la opt calculatoare învecinate.

Cablul transiverului se termină la placa de interfaţă din interiorul calculatorului. Placa de interfaţă conţine un cip controlor care transmite cadre către transiver şi recepţionează cadre de la acesta. Controlorul este responsabil cu asamblarea datelor în formatul de cadru corespunzător, precum şi cu calculul sumelor de control pentru cadrele trimise şi verificarea lor pentru cadrele primite. Unele cipuri controlor gestionează şi un set de zone tampon pentru cadrele primite, o coadă de zone tampon pentru transmisie, transferurile DMA cu calculatoarele gazdă şi alte aspecte legate de administrarea reţelei.

La 10Base2, conexiunea cu cablul se face printr-un conector BNC pasiv cu joncţiune în T. Electronica transiverului este pe placa controlorului şi fiecare staţie are întotdeauna propriul transiver.

La 10Base-T, nu există nici un cablu, ci doar un concentrator – o cutie plină de electronică. Adăugarea sau îndepărtarea unei staţii este mai simplă în această configuraţie, iar întreruperile cablului pot fi detectate uşor. Dezavantajul lui 10base-T este acela că dimensiunea maximă a cablului care pleacă de la concentrator este de numai 100 de metri, poate chiar 150 de metri, dacă sunt folosite perechi torsadate de foarte bună calitate (categoria 5). De asemenea, un concentrator mare costă mii de dolari. Totuşi, 10Base-T devine tot mai popular

datorită uşurinţei de întreţinere. O versiune mai rapidă de 10Base-T (100Base-T) va fi discutată mai târziu în acest capitol.

A patra opţiune de cablare pentru 802.3 este **10Base-F**, care foloseşte fibre optice. Această alternativă este scumpă datorită costului conectorilor şi a terminatorilor, dar are o imunitate excelentă la zgomot şi este metoda care este aleasă atunci când transmisia se face între clădiri sau concentratoare aflate la distanţe mari. Sunt permise distante de kilometri. Oferă de asemenea o securitate bună, deoarece interceptarea traficului de pe o fibră de sticlă este mult mai dificil decât ascultarea traficului pe cablul de cupru.

Fig. 4-15 arată diferite moduri de cablare a unei clădiri. În fig. 4-15(a), un singur cablu este şer- puit din cameră în cameră, fiecare staţie fiind conectată direct la el în punctul cel mai apropiat. În fig. 4-15(b), o coloană verticală suie de la parter până la acoperiş, cu cabluri orizontale conectate direct la ea la fiecare etaj prin amplificatoare speciale (repetoare). În unele clădiri, cablurile orizontale sunt subţiri, iar coloana este groasă. Cea mai generală topologie este cea de arbore, ca în fig. 4- 15(c), deoarece o reţea cu două căi între unele perechi de staţii poate suferi din cauza interferenţei dintre cele două semnale.

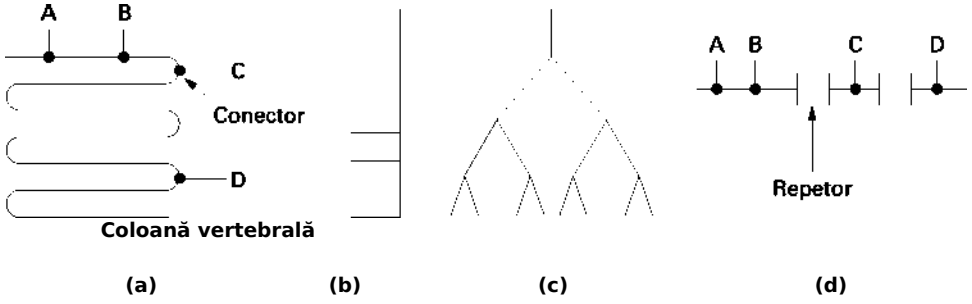


Fig. 4-15. Topologii de cablu. (a) Liniar. (b) Coloană. (c) Arbore. (d) Segmentat.

Fiecare versiune de 802.3 are o lungime maxim admisă de cablu pe segment. Pentru a permite reţele mai mari, mai multe cabluri pot fi conectate prin **repetoare** (**repeaters**), aşa cum se arată în fig. 4-15(d). Un repetor este un dispozitiv de nivel fizic. El recepţionează, amplifică şi retransmite semnale în ambele direcţii. În ceea ce priveşte programarea, o serie de segmente de cablu conectate prin repetoare nu prezintă nici o diferenţă faţă de un singur cablu (cu excepţia unei oarecare întârzieri introduse de repetoare). Un sistem poate conţine segmente de cablu multiple şi repetoare multiple, dar două transivere nu pot fi la o distanţă mai mare de 2,5 km şi nici o cale între oricare două transivere nu poate traversa mai mult de 4 repetoare.

4.3.2 Codificarea Manchester

Nici una din versiunile lui 802.3 nu foloseşte o codificare binară directă, cu 0 volţi pentru un bit 0 şi 5 volţi pentru un bit 1, deoarece aceasta conduce la ambiguităţi. Dacă o staţie trimite şirul de biţi 00010000, altele l-ar putea interpreta fals ca 10000000 sau 01000000 întrucât nu pot distinge diferenţa între un emiţător

inactiv (0 volţi) şi un bit 0 (0 volţi). Această problemă poate fi rezolvată prin utilizarea valorilor +1V pentru 1 şi -1V pentru 0. Totuşi, această soluţie nu rezolvă problema receptorului care va eşantiona semnalul cu o frecvenţă uşor diferită de cea pe care emiţătorul o foloseşte ca să-l genereze. Ceasurile diferite pot duce la o desincronizare între emiţător şi receptor în ceea ce priveşte graniţele biţilor, în special după un şir lung de 0 consecutivi sau de 1 consecutivi.

Ceea ce le trebuie receptorilor este un mijloc de a determina fără dubii începutul, sfârşitul şi jumătatea fiecărui bit fără ajutorul unui ceas extern. Două astfel de abordări se numesc **codificarea Manchester** (**Manchester encoding**) şi **codificarea Manchester diferenţială** (**differential Manchester encoding**). În cazul codificării Manchester, fiecare perioadă a unui bit este împărţită în două intervale egale. Un bit 1 este trimis stabilind un voltaj ridicat în timpul primului interval şi scăzut în cel de-al doilea. Un 0 binar este trimis exact invers: întâi nivelul scăzut iar apoi cel ridicat. Această strategie asigură că fiecare perioadă a unui bit are o tranziţie la mijloc, uşurând sincronizarea între emiţător şi receptor. Un dezavantaj al codificării Manchester este acela că necesită o lărgime de bandă dublă faţă de codificarea binară directă, deoarece impulsurile au durata pe jumătate. Codificarea Manchester este prezentată în fig. 4-16(b).

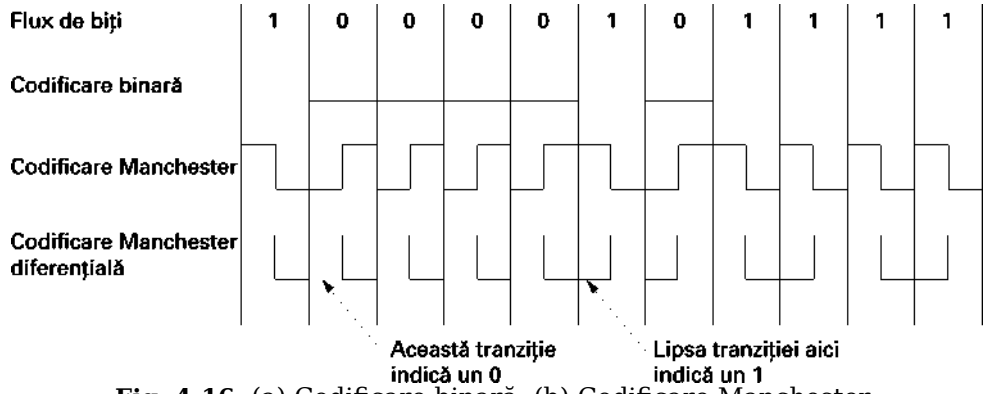

Flux de biţi
1 0 0 0 0 1 0 1 1 1 1
Codificare binară
Codificare Manchester
Codificare Manchester diferenţială
Această tranziţie indică un 0
Lipsa tranziţiei aici indică un 1

Fig. 4-16. (a) Codificare binară. (b) Codificare Manchester. (c) Codificare Manchester diferenţială.

Codificarea Manchester diferenţială, prezentată în fig. 4-16(c), este o variantă a codificării Manchester clasice. În aceasta, un bit 1 este indicat prin absenţa tranziţiei la începutul unui interval. Un bit 0 este indicat prin prezenţa unei tranziţii la începutul intervalului. În ambele cazuri, există şi o tranziţie la mijloc. Strategia diferenţială necesită un echipament mai complex, dar oferă o mai bună imunitate la zgomot. Toate sistemele 802.3 în banda de bază folosesc codificarea Manchester datorită simplităţii sale. Semnalul înalt este de +0.85 volţi iar semnalul scăzut este de -0.85 volţi, dând o valoare în curent continuu de 0 volţi. Ethernet nu foloseşte codificarea Manchester diferenţială, dar alte LAN-uri (de exemplu: 802.5 – LAN-urile de tip jeton pe inel) o folosesc.

4.3.3 Protocolul subnivelului MAC Ethernet

Structura cadrului original DIX (DEC, Intel, Xerox) este prezentată în fig. 4-17(a). Fiecare cadru începe cu un *Preambul* (Preamble) de 8 octeţi, fiecare octet conţinând şablonul de biţi 10101010. Codificarea Manchester a acestui şablon furnizează o undă dreptunghiulară de 10 MHz timp de 6.4 ps pentru a permite ceasului receptorului să se sincronizeze cu cel al emiţătorului. Ceasurile trebuie să rămână sincronizate pe durata cadrului, folosind codificarea Manchester pentru a detecta graniţele biţilor.

Cadrul conţine două adrese, una pentru destinaţie şi una pentru sursă. Standardul permite adrese pe 2 şi pe 6 octeţi, dar parametrii definiţi pentru standardul în banda de bază de 10 Mbps folosesc numai adrese pe 6 octeţi. Bitul cel mai semnificativ al adresei destinaţie este 0 pentru adresele obişnuite şi 1 pentru adresele de grup. Adresele de grup permit mai multor staţii să asculte de la o singură adresă. Când un cadru este trimis la o adresă de grup, toate staţiile din grup îl recepţionează. Trimiterea către un grup de staţii este numită **multicast** (**trimitere multiplă**). Adresa având toţi biţii 1 este rezervată pentru **broadcast** (**difuzare**). Un cadru conţinând numai biţi de 1 în câmpul destinaţie este distribuit tuturor staţiilor din reţea. Diferenţa dintre trimitere multiplă şi difuzare este suficient de importantă ca să merite a fi repetată: un cadru de trimitere multiplă este trimis unui grup de staţii selectate pe Ethernet; un cadru de difuzare este trimis tuturor staţiilor de pe Ethernet. Deci, trimiterea multiplă este mai selectivă, dar implică gestiunea grupurilor. Difuzarea este mai imprecisă dar nu necesită nici un fel de gestiune de grup.

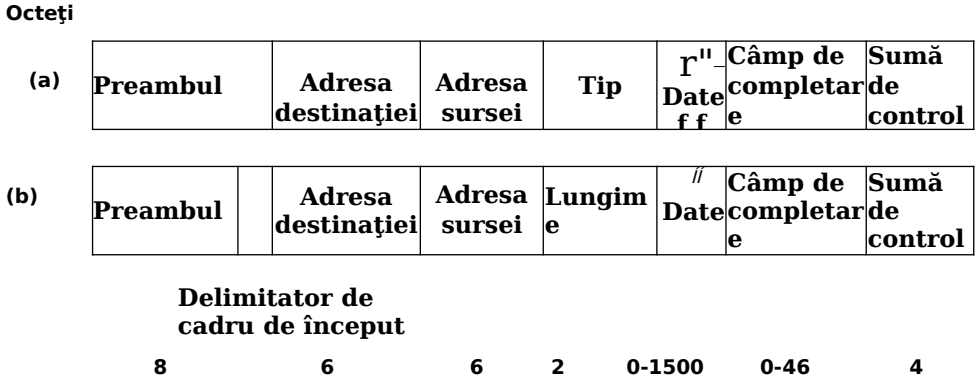


Fig. 4-17. Formatul cadrelor. (a) DIX Ethernet. (b) IEEE 802.3.

O altă trăsătură interesantă a adresării este utilizarea bitului 46 (vecin cu cel mai semnificativ bit) pentru a distinge adresele locale de cele globale. Adresele locale sunt stabilite de fiecare administrator de reţea şi nu au semnificaţie în afara reţelei locale. În schimb, adresele globale sunt asignate de IEEE pentru a se asigura că oricare două staţii din lume nu au aceeaşi adresă globală. Cu 48 – 2 = 46 biţi disponibili, există aproximativ $7x10^{13}$ adrese globale. Ideea este că orice staţie poate adresa în mod unic orice altă staţie specificând numai numărul corect pe 48 de biţi. Este sarcina nivelului reţea să-şi dea seama cum să localizeze destinatarul.

În continuare urmează câmpul „Tip” (Type), care îi spune receptorului ce să facă cu cadrul. Numeroase protocoale de nivel reţea pot fi folosite simultan pe aceeaşi maşină, astfel încât, atunci când un cadru Ethernet ajunge, nucleul trebuie să ştie cui să-i trimită cadrul. Câmpul „Tip” specifică procesul căruia îi este destinat cadrul.

Apoi urmează datele, până la 1500 de octeţi. Această limită a fost aleasă oarecum arbitrar la momentul în care standardul DIX a fost solidificat, în special din cauza consideraţiei că un transiver are nevoie de suficient RAM ca să conţină un cadru întreg şi RAM era scumpă în 1978. O valoare mai mare pentru această limită ar fi însemnat mai mult RAM, deci un transiver mai scump.

În afară de faptul că există o lungime maximă a cadrelor, există şi o lungime minimă a cadrelor. Deşi un câmp de date de 0 octeţi este uneori util, el poate duce la o situaţie problemă. Când un transiver detectează o coliziune, el trunchiază cadrul curent, ceea ce înseamnă că fragmente răzleţe de cadre şi biţi rătăciţi apar mereu pe cablu. Pentru a facilita distingerea cadrelor valide de reziduuri, Ethernet cere ca toate cadrele valide să aibă cel puţin 64 de octeţi, incluzând adresa destinaţiei şi suma de control. Dacă porţiunea de date dintr-un cadru este mai mică de 46 de octeţi, se foloseşte câmpul de completare pentru a se ajunge la lungimea minimă necesară.

Un alt motiv (şi mai important) de a avea o lungime minimă a cadrului este de a preveni situaţia în care o staţie termină transmisia unui cadru scurt înainte ca primul bit să ajungă la capătul cel mai îndepărtat al cablului, unde poate intra în coliziune cu un alt cadru. Această problemă este ilustrată în fig. 4-18. La momentul 0, staţia *A,* aflată la un capăt al reţelei, expediază un cadru. Să notăm cu τ timpul de propagare al cadrului până la celălalt capăt. Exact înainte de sosirea cadrului la celălalt capăt (adică la momentul τ – e), cea mai îndepărtată staţie faţă de *A*, staţia *B,* începe să transmită. Când *B* observă că primeşte mai multă putere decât emite, ştie că a apărut o coliziune, prin urmare abandonează transmisia şi generează o rafală de 48 de biţi de zgomot pentru a avertiza toate celelalte staţii. Aproximativ la momentul 2τ, emiţătorul observă apariţia zgomotului şi îşi abandonează la rândul său transmisia. Apoi aşteaptă un timp aleatoriu înainte de a încerca din nou.

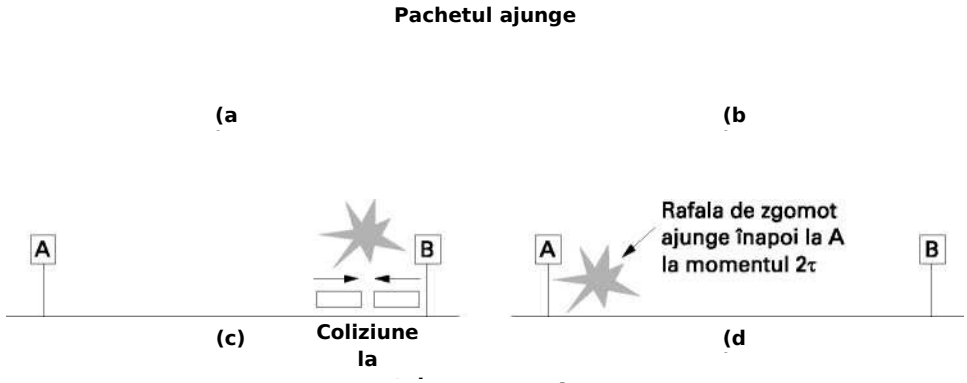


Fig. 4-18. Detectarea coliziunii poate

Dacă o staţie încearcă să transmită un cadru foarte scurt, este posibil să apară o coliziune, dar transmisia se termină înainte ca zgomotul produs să se întoarcă la momentul 2τ. Emiţătorul va ajunge incorect la concluzia că transmisia cadrului s-a încheiat cu succes. Pentru a preveni apariţia acestei situaţii, transmisia fiecărui

cadru trebuie să ia mai mult de 2τ. Pentru un LAN la 10 Mbps cu o lungime maximă de 2500 metri şi patru repetoare (conform specificaţiei 802.3), durata unei călătorii dus-întors (incluzând şi timpul necesar propagării prin cele 4 repetoare) a fost calculată la aproximativ 50 ps în cel mai defavorabil caz – inclusiv timpul trecerii prin repetoare, care în mod sigur nu este zero. Prin urmare, transmisia unui cadru minim trebuie să dureze cel puţin 50 ps pentru a se transmite. La 10 Mbps, un bit durează 100 ns, astfel încât cel mai mic cadru trebuie să aibă 500 de biţi pentru o funcţionare garantată. Pentru a adăuga un oarecare spaţiu de siguranţă, acest număr a fost

mărit la 512 biţi, adică 64 de octeţi. Cadrele cu mai puţin de 64 de octeţi utili sunt completate până la 64 de octeţi folosind câmpul de completare.

Pe măsură ce viteza reţelelor creşte, lungimea minimă a cadrului trebuie să crească sau lungimea maximă a cablului trebuie să scadă proporţional. Pentru un LAN de 2500 de metri operând la 1 Gbps, dimensiunea minimă a cadrului ar trebui să fie de 6400 de octeţi. Alternativ, dimensiunea minimă a cadrului ar putea fi de 640 octeţi, iar distanţa maximă între două staţii de 250 de metri. Aceste restricţii devin din ce în ce mai neplăcute pe măsură ce ne îndreptăm spre reţele cu viteze de ordinul gigabiţilor.

Ultimul câmp la 802.3 este *Suma de control (Checksum).* Aceasta este de fapt un cod de dispersie pe 32 de biţi (32-bit hash-code) a datelor. Dacă anumiţi biţi de date sunt recepţionaţi eronat (datorită zgomotului de pe cablu), suma de control va fi aproape sigur greşită şi va fi detectată o eroare. Algoritmul sumei de control este un control cu redundanţă ciclică de tipul celui discutat în cap. 3. El realizează doar detectarea erorilor şi nu are legătură cu corectarea lor.

Când IEEE a standardizat Ethernetul, comitetul a decis două schimbări la formatul DIX, după cum se vede în fig. 4-17(b). Prima a fost reducerea preambulului la 7 octeţi, folosind ultimul octet ca un delimitator de cadru iniţial („Start of Frame”) pentru compatibilizarea cu 802.4 şi 802.5. A doua schimbare a constat în transformarea câmpului „tip” într-un câmp „lungime”. Desigur, acum receptorul nu mai ştia ce să facă cu un cadru care sosea, dar această problemă a fost rezolvată prin adăugarea unui mic antet porţiunii de date, pentru a oferi această informaţie. Vom discuta formatul porţiunii de date când ajungem la controlul legăturilor logice, mai târziu în acest capitol.

Din păcate, la momentul publicării lui 802.3, se utilizau deja dispozitive hardware şi aplicaţii software pentru DIX Ethernet, astfel încât producătorii şi utilizatorii nu prea erau entuziaşti să convertească câmpul „tip” în câmpul „lungime”. În 1997 IEEE a capitulat şi a declarat că ambele standarde erau acceptabile. Din fericire, toate câmpurile „tip” folosite înainte de 1997 erau mai mari de 1500. Prin urmare, orice număr s-ar afla în acea poziţie care este mai mic sau egal cu 1500 poate fi interpretat ca „lungime”, iar orice număr mai mare decât 1500 poate fi interpretat ca „tip”. Acum IEEE poate susţine că fiecare îi foloseşte standardul şi toată lumea poate să îşi vadă de treabă făcând ce făceau şi înainte, fără să aibă remuşcări.

4.3.4 Algoritmul de regresie exponenţială binară

Să vedem acum algoritmul prin care se generează timpii aleatorii atunci când apare o coliziune. Modelul este cel din fig. 4-5. După o coliziune, timpul este împărţit în intervale discrete, a căror lungime este egală cu timpul de propagare dus-întors prin mediu în cazul cel mai defavorabil (2τ). Pentru a se potrivi cu cea mai lungă cale permisă de 802.3 (2.5 km şi patru repetoare), mărimea cuantei a fost fixată la 512 intervale de bit, adică 51.2 ps – după cum a fost menţionat anterior.

După prima coliziune, fiecare staţie aşteaptă fie 0, fie 1 cuante înainte să încerce

din nou. Dacă două staţii intră în coliziune şi fiecare alege acelaşi număr aleatoriu, vor intra din nou în coliziune. După a doua coliziune, fiecare aşteaptă la întâmplare 0, 1, 2 sau 3 cuante. Dacă se produce o a treia coliziune (probabilitatea este de 0.25), atunci, data viitoare, numărul de cuante aşteptate va fi ales aleatoriu din intervalul de la 0 la $2^3 - 1$.

În general, după *i* coliziuni, se aşteaptă un număr aleatoriu de cuante între 0 şi $2^i - 1$. Oricum, după un număr de 10 coliziuni, intervalul de aşteptare este îngheţat la un maxim de 1023 de cuante. După 16 coliziuni, controlorul aruncă prosopul‡‡‡‡ şi raportează eşec calculatorului. Recuperarea ulterioară din situaţia de eroare cade în sarcina nivelurilor superioare.

Acest algoritm, numit **algoritmul de regresie exponenţială binară** (**binary exponential backoff algorithm**), a fost conceput să se poată adapta dinamic la numărul staţiilor care încearcă să transmită. Dacă intervalul de generare aleatorie a fost pentru toate coliziunile 1023, şansa ca 2 staţii să intre în coliziune pentru a doua oară este neglijabilă, dar timpul mediu de aşteptare după o coliziune ar fi de sute de cuante, introducând o întârziere semnificativă. Pe de altă parte, dacă fiecare staţie aşteaptă mereu sau zero sau o cuantă, atunci dacă 100 de staţii ar încerca să transmită deodată, ele ar intra în coliziune iar şi iar, până când 99 dintre ele aleg 0 şi una 1 sau invers. Aceasta ar putea dura ani de zile. Lăsând intervalul de generare aleatorie să crească exponenţial pe măsură ce apar tot mai multe coliziuni, algoritmul asigură o întârziere minimă când se ciocnesc numai câteva staţii, dar garantează de asemenea că ciocnirea este rezolvată într-un interval rezonabil atunci când este vorba de mai multe staţii. Limitarea intervalului la 1023 de cuante previne creşterea peste măsura a întârzierilor.

Aşa cum am arătat până acum, CSMA/CD nu oferă confirmări. Cum simpla absenţă a coliziunilor nu garantează că biţii nu au fost modificaţi de zgomotul de pe cablu, pentru o comunicaţie sigură, destinaţia trebuie să verifice suma de control şi, dacă este corectă, să trimită înapoi către sursă un cadru de confirmare. În mod normal, din punct de vedere al protocolului, această confirmare ar fi doar un alt cadru de date şi ar trebui să lupte pentru timp de canal, ca orice cadru de date. Totuşi, cu o simplă modificare a algoritmului de tratare a conflictelor s-ar permite o confirmare rapidă a recepţionării cadrului (Tokoro şi Tamaru, 1977): prima cuantă de conflict care urmează unei transmisii cu succes ar trebui rezervată pentru staţia destinaţie. Din nefericire, standardul nu oferă această posibilitate.

‡‡‡‡Aşa procedează antrenorul unui boxer când hotărăşte ca acesta să abandoneze lupta.

4.3.5 Performanţele Ethernet-ului

Să examinăm pe scurt performanţele standardului 802.3 în condiţii de încărcare mare şi constantă, dată de *k* staţii gata mereu să transmită. O analiză riguroasă a algoritmului de regresie exponenţială binară ar fi complicată. În schimb vom proceda ca Metcalfe şi Boggs (1976) şi vom presupune o probabilitate de retransmisie constantă pentru fiecare cuantă. Dacă fiecare staţie transmite în timpul unei cuante de conflict cu probabilitatea *p,* probabilitatea *A* ca o staţie să primească canalul în această cuantă este:

$$A = kp(1 - p)^{k-1}$$

A este maxim când $p = 1/k$, şi *A* *1/e* atunci când *k* *<x>.*
Probabilitatea ca intervalul de conflict
să aibă exact *j* cuante este $A\ (1 - A)^{j-1,}$ astfel că numărul mediu de cuante pe conflict este dat de:

$$\sum_{j=0} jAV - A))^{1} = A$$

Întrucât fiecare cuantă durează 2τ, intervalul de conflict mediu, *w,* este $2\tau/A$. Presupunând *p* optim, numărul mediu de cuante de conflict nu este niciodată mai mare decât *e,* deci *w* este cel mult *2ze* ® 5.4τ.

Dacă pentru a transmite un cadru de lungime medie sunt necesare P secunde, atunci când multe staţii au cadre de transmis se obţine:

$$\text{Eficienţa} \qquad \frac{P}{P + 2\tau /} \qquad (4-$$

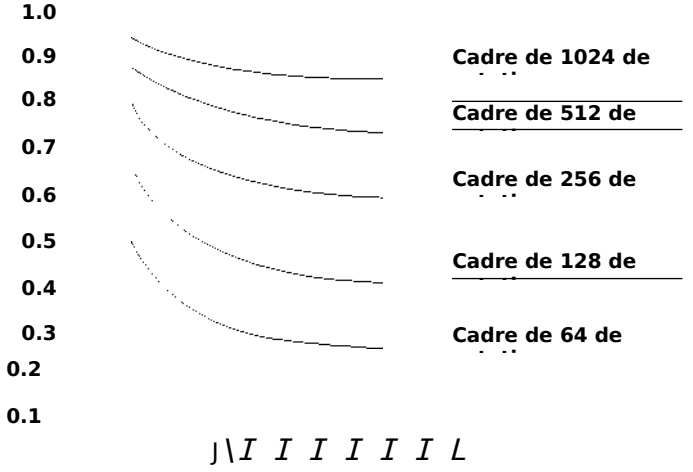
Aici vedem cum lungimea maximă a cablului dintre oricare două staţii influenţează calculul performanţelor, sugerând şi alte topologii decât cea din fig. 4-15(a). Cu cât cablul este mai lung, cu atât intervalul de conflict este mai lung. Acesta este motivul pentru care standardul Ethernet specifică o lungime maximă a cablului.

Este instructiv să formulăm ecuaţia (4-6) şi în termeni de lungime de cadru *F,* lărgime de bandă a reţelei *B,* lungime a cablului *L* şi viteză de propagare a semnalului *c,* pentru cazul optim cu *e* cuante de conflict pe cadru. Cu $P = F/B$, ecuaţia (4-6) devine:

$$\text{Eficienţa} \qquad \frac{1}{1 + 2BLe /} \qquad (4-$$

Atunci când al doilea termen al numitorului este mare, eficienţa reţelei va fi mică. Mai precis, creşterea lărgimii de bandă sau a distanţei (produsul *BL)* reduce eficienţa pentru o lungime dată a cadrului. Din nefericire, o mare parte din cercetarea în domeniul hardware-ului de reţea a ţintit exact creşterea acestui produs. Oamenii doresc lărgime de bandă mare pe distanţe lungi (de exemplu, MAN-urile cu fibră optică), ceea ce sugerează că Ethernetul implementat în acest fel poate să nu fie cel mai bun sistem pentru aceste aplicaţii. Vom vedea alte modalităţi de a implementa Ethernet când ajungem la Ethernetul comutat mai târziu în acest capitol.

În fig. 4-19 este trasată eficienţa canalului în funcţie de numărul staţiilor gata de transmisie, pentru 2τ=51.2 ps şi o rată de transmisie a datelor de 10 Mbps, folosind ecuaţia (4-7). Cu o mărime a cuantei de 64 de octeţi, nu este surprinzător faptul că nu sunt eficiente cadrele de 64 de octeţi. Pe de altă parte, cu cadre de 1024 de octeţi şi o valoare asimptotică de *e* cuante de 64 de octeţi pe interval de conflict, perioada de conflict este de 174 de octeţi, iar eficienţa este 0.85.



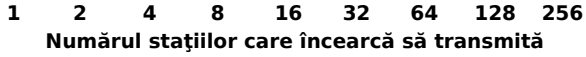


Fig. 4-19. Eficienţa 802.3 la 10 Mbps cu dimensiunea cuantelor de 512 biţi.

Pentru a determina numărul mediu de staţii gata de transmisie în condiţiile unei încărcări mari, putem să ne folosim de următoarea observaţie (brută). Fiecare cadru acaparează canalul pentru o perioadă de conflict şi un interval de transmisie a unui cadru, totalizând un timp de $P + w$ secunde. Prin urmare, numărul de cadre pe secundă este $1/(P + w)$. Dacă fiecare staţie generează cadre cu o rată medie de X cadre/sec, atunci când sistemul este în starea *k*,* rata totală de intrare combinată a tuturor staţiilor neblocate este de *kk* cadre/sec. Deoarece la echilibru ratele de intrare şi de ieşire trebuie să fie identice, putem egala aceste două expresii şi putem rezolva pentru k (nu uitaţi că w este funcţie de k). O analiză mai sofisticată este dată în (Bertsekas şi Gallager,1992).

Probabil că merită să menţionăm că s-au realizat numeroase analize teoretice ale performanţelor pentru Ethernet (şi pentru alte reţele). De fapt, toată această muncă a presupus că traficul este de tip Poisson. Pe măsură ce cercetătorii au început să se uite la datele reale, s-a descoperit că traficul în reţea este rareori Poisson, în schimb este autosimilar (Paxson şi Floyd,1994; şi Willinger ş.a., 1995). Aceasta înseamnă că nici prin calcularea valorilor medii pe perioade lungi de timp nu se obţine o netezire a traficului. Altfel spus, numărul mediu de pachete în fiecare minut al unei ore variază la fel de mult ca şi numărul mediu de pachete în fiecare secundă a unui minut. Consecinţa acestei descoperiri este că majoritatea modelelor de trafic în reţea nu se aplică lumii reale şi ar trebui luate cu un pic (sau, mai bine, cu o tonă) de sare!

4.3.6 Ethernetül comutat

Pe măsură ce la Ethernet sunt adăugate tot mai multe staţii, traficul va creşte. În cele din urmă, LAN-ul se va satura. O cale de ieşire din această situaţie este mărirea vitezei, să zicem, de la 10 Mbps la 100 Mbps. Dar, odată cu creşterea în importanţă a aplicaţiilor multimedia, chiar un Ethernet de 100 Mbps sau l-Gbps poate deveni saturat.

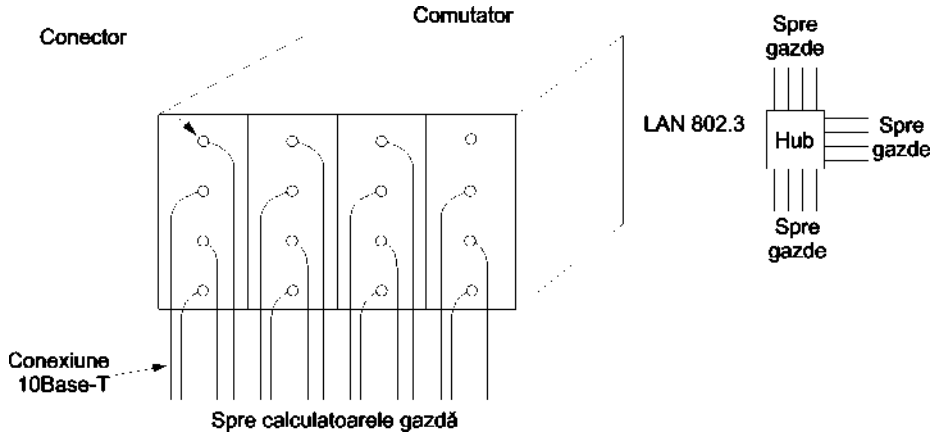


Fig. 4-20. Un LAN 802.3 comutat.

Din fericire, este posibilă o soluţie diferită, mai puţin drastică: un Ethernet comutat ca cel din fig. 4-20. Inima acestui sistem este un comutator care conţine o placă de bază (similară unui fund de sertar – backplane) de mare viteză şi, în general, loc pentru 4 până la 32 de plăci de reţea plug-în,

---

k staţii gata de transmisie fiecare având între 1 şi 8 conectori. Cel mai des, fiecare conector are o conexiune prin perechi torsadate de tip 10Base-T cu un singur calculator gazdă.

Atunci când o staţie doreşte să transmită un cadru 802.3, trimite un cadru standard către comutator. Placa plug-în care primeşte cadrul verifică dacă el este destinat pentru una din celelalte staţii conectate la aceeaşi placă. Dacă da, cadrul este copiat acolo. Dacă nu, cadrul este trimis prin placa de bază a comutatorului (backplane) către placa staţiei destinaţie. Placa de bază a comutatorului rulează în mod obişnuit la peste 1 Gbps folosind protocolul proprietar.

Ce se întâmplă dacă două calculatoare legate la aceeaşi placă plug-în transmit cadre în acelaşi timp? Depinde de cum a fost construită placa. O posibilitate este ca toate porturile de pe placă să fie legate împreună pentru a forma un LAN local pe placă. Coliziunile din acest LAN pe placă vor fi detectate şi tratate la fel ca orice altă coliziune dintr-o reţea CSMA/CD – cu retransmisii utilizând algoritmul de regresie binară. Cu acest tip de placă plug-în este posibilă o singură transmisie pe placă la un moment dat, dar toate plăcile pot transmite în paralel. Astfel concepute, fiecare dintre plăci îşi formează propriul **domeniu de coliziune** (**collision domain**), independent de celelalte.

La celălalt tip de placă plug-în, fiecare port de intrare utilizează un registru tampon, astfel încât cadrele care vin sunt stocate în memoria RAM inclusă în placă, pe măsură ce sosesc. Această concepţie permite tuturor porturilor de intrare să recepţioneze (şi să transmită) cadre în acelaşi timp, pentru operare duplex integral (full duplex), în paralel. Odată ce un cadru a fost recepţionat în întregime, placa poate verifica dacă el este destinat pentru un alt port de pe aceeaşi placă, sau

pentru un port aflat la distanţă. În primul caz, el poate fi transmis direct la destinaţie. În cel de-al doilea, el trebuie transmis prin placa de bază a comutatorului către placa corespunzătoare. În acest mod, fiecare port este un domeniu de coliziune separat, deci nu se mai produc coliziuni. Adesea, productivitatea întregului sistem poate fi îmbunătăţită astfel cu un ordin de mărime faţă de 10Base-5, care are un singur domeniu de coliziune pentru întreg sistemul.

Întrucât comutatorul stă şi aşteaptă cadre standard Ethernet pe fiecare port de intrare, putem folosi unele porturi drept concentratori. În fig. 4-20, portul din colţul din dreapta sus este conectat nu la o singură staţie, ci la un concentrator cu 12 porturi. Pe măsură ce cadrele sosesc la concentrator, ele concurează pentru canale în mod obişnuit, cu apariţie de coliziuni şi algoritm de regresie binară. Cadrele transmise cu succes ajung la comutator, unde sunt tratate ca orice cadru de intrare: sunt îndreptate către linia de ieşire corectă prin placa de bază de viteză mare. Concentratoarele sunt mai ieftine decât comutatoarele, dar, datorită preţurilor în scădere ale comutatoarelor, ele ies treptat din uz. Totuşi, mai există concentratoare rămase moştenire.

4.3.7 Ethernet-ul rapid

La început, 10 Mbps păreau raiul pe pământ, la fel cum modemurile 1200-bps păreau divine utilizatorilor modemurilor acustice de 300 bps. Totuşi, noutatea s-a uzat rapid. Ca un fel de corolar al Legii lui Parkinson („Munca se dilată astfel încât să ocupe tot timpul aflat la dispoziţie”), se părea că datele se dilată pentru a umple lărgimea de bandă disponibilă. Pentru a creşte viteza, diverse grupuri industriale au propus două noi LAN-uri optice bazate pe două inele. Una era numită FDDI (Fiber Distributed Data Interface, rom: Interfaţă de Date Distribuită pe Fibră), cealaltă se numea canal de fibră (Fibre Channel[1]). Pentru a scurta o poveste lungă, amândouă au fost folosite ca şi reţele de coloană vertebrală şi nici una nu a reuşit să ajungă în birourile utilizatorilor finali. În ambele cazuri ma-

[1] Este denumit „fibre channel” şi nu „fiber channel”; autorul documentului era englez

nagementul staţiilor era prea complicat, ceea ce ducea la cip-uri complexe şi preţuri ridicate. Lecţia care trebuia învăţată de aici era KISS (Keep it simple, Stupid, rom: Lasă lucrurile simple, prostule).

În orice caz, eşecul LAN-urilor optice în a se impune pe piaţă a lăsat un gol în care au înflorit o varietate de Ethernet-uri la viteze de peste 10 Mbps. Multe instalaţii aveau nevoie de mai multă lărgime de bandă şi prin urmare aveau numeroase LAN-uri de 10 Mbps conectate printr-un labirint de repetoare, punţi, rutere şi porţi, deşi administratorilor de reţea li se părea mai degrabă că erau conectate cu gumă de mestecat şi resturi de sârmă.

În acest mediu IEEE a reconvocat comitetul 802.3 în 1992 cu instrucţiuni de a produce un LAN mai rapid. O propunere a fost aceea de a păstra 802.3 exact cum era, dar să-l facă să meargă mai repede. O altă propunere era să-l refacă total astfel încât să îi ofere o mulţime de noi proprietăţi, cum ar fi trafic în timp real şi voce digitizată, dar să păstreze vechiul nume (din raţiuni de marketing). După ceva

confruntări, comitetul a decis să păstreze 802.3 aşa cum era, dar să-l facă mai rapid. Cei care susţinuseră propunerea înfrântă au făcut ceea ce orice indivizi din industria de calculatoare ar fi făcut în aceste circumstanţe – s-au detaşat şi au format propriul lor comitet care a standardizat LAN-ul (în ceea ce va fi versiunea 802.12). Încercarea lor a eşuat lamentabil.

Comitetul 802.3 a decis să continue cu un Ethernet ameliorat din trei motive principale:

1. Nevoia de a fi compatibil retroactiv cu LAN-urile Ethernet existente;
2. Teama că un nou protocol ar putea avea consecinţe negative neprevăzute;
3. Dorinţa de a termina treaba înainte ca tehnologia să se schimbe.

Munca a fost făcută rapid (după standardele comitetului), iar rezultatul, 802.3u, a fost aprobat oficial de IEEE în iunie 1995. Din punct de vedere tehnic, 802.3u nu este un standard nou, ci o adăugire la standardul 802.3 existent (pentru a accentua compatibilitatea cu versiunile anterioare). Din moment ce toată lumea îl denumeşte **Ethernet rapid** (**Fast Ethernet**), în loc de 802.3u, îl vom denumi şi noi la fel.

Ideea de bază din spatele Ethernetului rapid era simplă: păstrează vechile formate de cadre, interfeţele şi regulile procedurale, dar reduce durata bitului de la 100 ns la 10 ns. Din punct de vedere tehnic, ar fi fost posibil să copieze fie 10Base-5 sau 10Base-2 şi să detecteze în continuare coliziunile la timp pur şi simplu reducând lungimea maximă a cablului cu un factor de 10. Totuşi, avantajele cablării 10Base-T erau atât de copleşitoare, încât Ethernetul rapid este bazat în întregime pe acest design. Prin urmare, toate sistemele de Ethernet rapid folosesc concentratoare şi comutatoare; cabluri multipunct cu conectori vampir sau BNC nu sunt permise.

Totuşi, rămân câteva alegeri de făcut, dintre care cea mai importantă este ce tip de cabluri să fie suportate. Un concurent era cablul torsadat categoria 3. Argumentul pro era că practic fiecare birou în lumea occidentală are cel puţin patru cabluri răsucite categoria 3 (sau mai mult) care îl conectează cu un centru de conexiuni telefonice la cel mult 100 m distanţă. Uneori există două astfel de cabluri. Prin urmare, folosind cablurile torsadate categoria 3 ar fi făcut posibilă conectarea calculatoarelor de birou la Ethernet fără să fie necesară recablarea clădirii, un avantaj enorm pentru multe organizaţii.

Principalul dezavantaj al cablurilor torsadate categoria 3 este incapacitatea lor de a transmite semnale de 200 megabaud (100Mbps cu codificare Manchester) pe o lungime de 100 de metri, care este distanţa maximă de la calculator la concentrator specificată pentru 10Base-T (vezi fig. 4-13). Dimpotrivă, cablurile torsadate categoria 5 fac faţă uşor distanţelor de 100 m, iar fibra face faţă unor distanţe mult mai mari. Compromisul la care s-a ajuns a fost să permită toate trei posibilităţile, după cum reiese din fig. 4-21, ca să se îmbunătăţească soluţia de categorie 3 pentru a-i oferi capacitatea adiţională de transportare de care avea nevoie.

Nume	Cablu	Segment maxim	Avantaje
100Base-T4	Cablu torsadat	100 m	Foloseşte UTP categoria 3
100Base-TX	Cablu torsadat	100 m	Full duplex la 100 Mbps (UTP Cat 5)

100Base-FX	Fibră de sticlă	2000 m	Full duplex la 100 Mbps; distanţe

Fig. 4-21. Cablarea originală a Ethernet-ului rapid.

Schema de categorie 3 UTP, numită 100Base-T4, foloseşte o viteză de semnalizare de 25MHz, cu numai 25% mai rapid decât Ethernetul standard de 20MHz (amintiţi-vă de codificarea Manchester care, după cum reiese din fig. 4-16, necesită două rotaţii de ceas pentru fiecare dintre cei 10 milioane de biţi pe secundă). Totuşi, pentru a obţine lărgimea de bandă necesară, 100Base-T4 necesită patru perechi răsucite. Deoarece cablarea telefonică standard include de decenii patru perechi torsadate per cablu, majoritatea birourilor sunt capabile să facă faţă. Desigur, înseamnă să renunţi la telefonul din birou, dar acesta este un preţ mic pentru un e-mail mai rapid.

Din cele patru perechi torsadate una merge întotdeauna către concentrator, una vine de la concentrator, iar celelalte două sunt comutabile în direcţia transmisiunii curente. Codificarea Manchester nu poate fi folosită din cauza cerinţelor de lărgime de bandă, dar date fiind ceasurile moderne şi distanţele scurte, nici nu mai este necesară. În plus, sunt trimise semnale ternare, astfel încât în timpul unei singure rotaţii de ceas cablul poate conţine un 0, un 1 sau un 2. Având trei perechi torsadate în direcţia „înainte” şi cu semnalizare ternară, există 27 de simboluri posibile, şi deci se pot trimite 4 biţi cu o oarecare redundanţă. Transmiterea a 4 biţi în fiecare dintre cele 25 de milioane de rotaţii de ceas pe secundă oferă cei 100Mbps necesari. În plus, există întotdeauna un canal invers de 33.3Mbps care foloseşte perechea torsadată rămasă. Această schemă, cunoscută ca şi 8B/6T (8 biţi mapaţi pe 6 triţi), nu este cea mai elegantă din lume, dar funcţionează cu cablarea existentă.

Pentru cablarea de categorie 5, designul 100Base-TX este mai simplu deoarece cablurile fac faţă frecvenţelor de ceas de 125MHz. Numai 2 perechi torsadate sunt folosite – una către concentrator, şi alta dinspre el. Codificarea binară directă nu este folosită, ci în locul ei se află o schemă numită 4B/5B. Este preluată din FDDI şi este compatibilă cu el. Fiecare grup de cinci rotaţii de ceas, având fiecare una dintre cele două valori ale semnalului, generează 32 de combinaţii. 16 dintre acestea sunt folosite pentru a transmite grupurile de biţi 0000, 0001, 0010, ..., 1111. Din restul de 16, unele sunt folosite în scopuri de control, cum ar fi marcarea graniţelor cadrelor. Combinaţiile folosite au fost alese cu grijă, astfel încât să ofere suficiente tranziţii pentru a menţine sincronizarea ceasului. Sistemul 100Base-TX este integral duplex: simultan, staţiile pot transmite date la 100Mbps şi pot primi date la 100Mbps. Deseori oamenii se referă la 100Base-TX şi la 100Base-T4 cu denumirea comună 100Base-T.

Ultima opţiune, 100Base-Fx, foloseşte două linii de fibră multimod, una pentru fiecare direcţie, astfel încât sistemul este, de asemenea, integral duplex, cu 100Mbps în fiecare direcţie. În plus, distanţa dintre o staţie şi concentrator poate ajungă până la 2 km.

În 1997, comitetul a adăugat, la cerere, un nou tip de cablu, 100Base-T2, permiţând Ethernetului rapid să funcţioneze peste două perechi de cablu de categoria 3 deja existente. Totuşi, este nevoie de un procesor complicat de semnale

digitale pentru a face faţă schemelor de codificare, aşa că această opţiune este destul de scumpă. Până acum nu prea a fost utilizată, datorită complexităţii, costului, şi faptului că multe clădiri de birouri au fost deja recablate cu categoria 5 UTP.

100Base-T face posibile două tipuri de sisteme de interconectare: concentratoare şi comutatoare, după cum reiese din fig. 4-20. Într-un concentrator, toate liniile care sosesc (sau cel puţin toate liniile care ajung la o placă de extensie logică, formează un singur domeniu de coliziune. Toate regulile standard pot fi aplicate, incluzând algoritmul de regresie exponenţială binară, astfel încât sistemul funcţionează exact ca Ethernetul de modă veche. În particular, o singură staţie poate să transmită la un moment dat. Cu alte cuvinte, concentratoarele au nevoie de comunicaţii semi-duplex.

Într-un comutator, fiecare cadru care soseşte este ţinut într-o memorie tampon într-o placă de extensie şi transmis printr-o placă de bază de mare viteză de la placa sursă la placa destinaţie, dacă este nevoie. Această placă de bază a comutatorului nu a fost standardizată, şi nici nu trebuie să fie, din moment ce este cu desăvârşire ascunsă în interiorul comutatorului. Conform experienţelor precedente este foarte probabil că vânzătorii de comutatoare vor intra într-o concurenţă acerbă pentru a produce plăci de bază tot mai rapide şi pentru a îmbunătăţi performanţa sistemului. Deoarece cablurile 100Base-FX sunt prea lungi pentru algoritmul normal de coliziune, ele trebuie să fie conectate la comutatoare, astfel încât fiecare este un domeniu de coliziune distinct. Concentratoarele nu sunt permise în 100Base-FX.

Ca observaţie finală, practic toate comutatoarele pot face faţă unui mix de staţii 10 Mbps şi 100 Mbps, pentru a facilita modernizarea. Pe măsură ce un site obţine tot mai multe staţii de 100 Mbps, tot ceea ce trebuie să facă este să cumpere numărul necesar de plăci de extensie noi şi să le insereze în comutator. De fapt, standardul însuşi oferă o cale astfel încât două staţii să negocieze automat viteza optimă (10 sau 100Mbps) şi modul de comunicaţie (semi-duplex sau duplex integral). Majoritatea produselor de Ethernet rapid folosesc această caracteristică pentru a se autoconfigura.

4.3.8 Ethernetul Gigabit

De-abia se uscase cerneala pe standardul Ethernetului rapid când comitetul 802 a început să lucreze la un Ethernet şi mai rapid (1995). A fost numit imediat **Ethernet gigabit** (**Gigabit Ethernet**) şi a fost ratificat de IEEE în 1998 sub numele 802.3z. Această notaţie sugerează că Ethernetul gigabit va fi sfârşitul liniei, în afară de cazul în care cineva inventează rapid o nouă literă după z. Vom discuta mai jos câteva dintre caracteristicile de bază ale Ethernetului gigabit. Mai multe informaţii pot fi găsite în (Seifert, 1998). Scopurile comitetului 802.3z erau practic aceleaşi cu ale comitetului 802.3u: să facă Ethernetul de 10 ori mai rapid, astfel încât să rămână totuşi compatibil cu toate versiunile anterioare. În particular, Ethernetul gigabit trebuia să ofere suport pentru transferul fără confirmare a datagramelor atât pentru difuzare cât şi pentru trimitere multiplă, să folosească aceeaşi schemă

de adresare de 48 de biţi care era deja în uz, şi să menţină acelaşi format al cadrelor, inclusiv dimensiunile minime şi maxime ale acestora. Standardul final a reuşit să îndeplinească toate aceste scopuri.

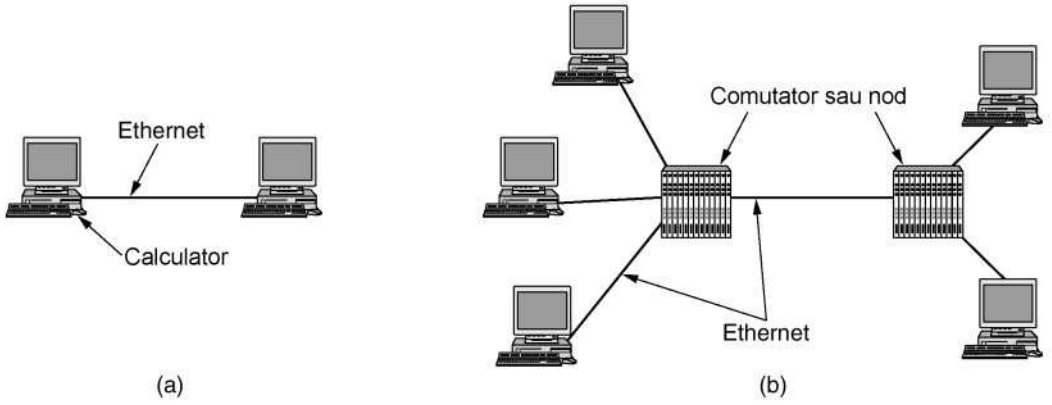


Fig. 4-22. (a) Un Ethernet cu două staţii. (b) Un Ethernet cu mai multe staţii.

Toate configuraţiile Ethernetului gigabit sunt punct-la-punct mai degrabă decât multipunct, ca şi în standardul original 10 Mbps, acum onorat cu denumirea de **Ethernet clasic**. În cea mai simpla configuraţie Ethernet, ilustrată în fig. 4-22(a), două calculatoare sunt conectate direct unul cu altul. Situaţia mai frecventă este totuşi aceea în care există un concentrator sau un comutator conectat la mai multe calculatoare, şi la alte concentratoare sau comutatoare adiţionale, ca în fig. 4-22(b). În ambele configuraţii, fiecare cablu individual de Ethernet conectează exact două sisteme – nici mai multe, nici mai puţine.

Ethernetul Gigabit suportă două moduri diferite de operare: modul duplex integral şi modul semi-duplex. Modul „normal” este cel duplex integral, care permite traficul în ambele direcţii în acelaşi timp. Acest mod este folosit atunci când există un comutator central la care sunt conectate calculatoarele (sau alte comutatoare) de la periferie. În această configuraţie, toate liniile sunt prevăzute cu spaţii tampon astfel încât fiecare calculator şi fiecare comutator sunt libere să transmită cadre oricând doresc. Emiţătorul nu trebuie să verifice canalul ca să vadă dacă este utilizat de altcineva, deoarece conflictele sunt imposibile. Pe linia dintre un calculator şi un comutator, calculatorul este singurul emiţător posibil către acel comutator şi transmisia va reuşi chiar şi în cazul în care comutatorul transmite în acelaşi timp un cadru către calculator, deoarece linia este duplex. Din moment ce conflictele sunt imposibile, protocolul CSMA/CD nu este utilizat, astfel încât lungimea maximă a cablului este determinată de argumente referitoare la intensitatea semnalului, şi nu de considerente referitoare la durata maximă a propagării zgomotului unei ciocniri către emiţător. Comutatoarele sunt libere să amestece şi să potrivească vitezele. Autoconfigurarea este suportată la fel ca în Ethernetul rapid.

Celălalt mod de operare, semi-duplex, este folosit când calculatoarele sunt conectate la un concentrator mai degrabă decât la un comutator. Un concentrator

nu stochează cadrele care vin într-un spaţiu tampon. În loc să facă asta, el conectează electric toate liniile în interior, simulând cablul multipunct folosit în Ethernetul clasic. În acest fel, există posibilitatea să apară coliziuni, astfel încât standardul CSMA/CD este necesar. Din cauză că un cadru de lungime minimă (adică de 64 de octeţi) poate fi transmis acum de 100 de ori mai rapid decât în Ethernetul clasic, distanţa maximă este de 100 de ori mai mică – adică de 25 de metri, pentru a menţine proprietatea esenţială că emiţătorul mai transmite încât atunci când zgomotul ajunge înapoi la el, chiar şi în cel mai rău caz. Cu un cablu lung de 2500 de metri, emiţătorul unui cadru de 64 de octeţi la 1Gbps va fi terminat de mult înainte ca drumul parcurs de cadru să fie măcar o zecime din cât are de mers – fără să mai socotim şi returul.

Comitetul 802.3z a considerat că o rază de 25 de metri este inacceptabilă şi a adăugat două caracteristici standardului pentru a mări raza. Prima caracteristică, numită **extinderea de către purtător**, se referă practic la a spune dispozitivului hardware să realinieze cadrul, mărindu-l până la 512 octeţi. Din moment ce această completare este adăugată de dispozitivul hardware emiţător şi este înlăturată de dispozitivul hardware receptor, partea software nu este conştientă de existenţa sa, şi prin urmare nu trebuie să sufere modificări. Desigur, transmiterea a 512 octeţi de lărgime de bandă pentru a transmite 46 octeţi de date ale utilizatorului (încărcătura propriu-zisă a cadrului de 64 de octeţi) are o eficienţă de transmitere de 9%.

A doua caracteristică, denumită **cadre în rafală** (**frame bursting)**, permite unui transmiţător să trimită o secvenţă concatenată de cadre multiple într-o singură transmisie. Dacă rafala totală este mai mică de 512 octeţi, dispozitivul hardware o completează din nou până la 512 octeţi. Dacă sunt destule cadre care aşteaptă să fie transmise, această schemă este foarte eficientă şi este preferată extinderii de către purtător. Aceste noi caracteristici extind raza la 200 de metri, ceea ce probabil este suficient pentru majoritatea birourilor.

Ca să fim sinceri, este destul de greu să ne imaginăm o organizaţie trecând prin toate dificultăţile cumpărării şi instalării plăcilor de Ethernet gigabit pentru a obţine o performanţă ridicată, şi apoi conectând calculatoarele printr-un concentrator pentru a simula Ethernetul clasic cu toate coliziunile sale. Deşi concentratoarele sunt oarecum mai ieftine decât comutatoarele, plăcile de Ethernet gigabit sunt totuşi scumpe. Să faci economii prin cumpărarea unui concentrator ieftin şi astfel să reduci performanţa noului sistem este o prostie. Totuşi, compatibilitatea cu versiunile anterioare este sacră în industria calculatoarelor, astfel încât comitetul 802.3z a trebuit să se conformeze.

Ethernetul gigabit suportă atât cablarea cu cupru cât şi cablarea cu fibră, precum este descris în fig. 4-23. Semnalizarea la nivelul de IGbps sau în jurul acestei viteze, înseamnă că sursa de lumină trebuie să fie închisă şi deschisă în mai puţin de 1ns. LED-urile pur şi simplu nu pot lucra atât de rapid, astfel încât este nevoie de lasere. Două lungimi de undă sunt permise: 0.85 microni (scurt) şi 1.3 microni (lung). Laserele de 0.85 microni sunt mai ieftine dar nu funcţionează pe fibra mono-mod.

Nume	Cablu	Segment maxim	Avantaje
1000Base-SX	Fibră de sticlă	550 m	Fibră multimod (50 şi 62,5 microni)
1000Base-LX	Fibră de sticlă	5000 m	Mono-mod (10p) sau multimod (50 şi 62,5 p)
1000Base-CX	2 perechi de STP	25 m	Pereche torsadată ecranată
1000Base-T	4 perechi de UTP	100 m	UTP Categoria 5

Fig. 4-23. Cablarea pentru Ethernet gigabit.

Sunt permise trei diametre de fibră: 10, 50 şi 62,5 microni. Prima este pentru mono-mod şi celelalte două sunt pentru multimod. Nu toate cele şase combinaţii sunt permise, totuşi, iar distanţa maximă depinde de combinaţia folosită. Numerele date în fig. 4-23 se referă la cazul cel mai fericit. În particular, 5000 de metri pot fi obişnuiţi numai dacă lasere de 1,3 microni operează pe fibră de 10 microni mono-mod, dar aceasta este cea mai bună alegere pentru structurile vertebrale din campusuri şi este de aşteptat să fie populară, deşi este şi cea mai scumpă alegere.

Opţiunea 1000Base-CX foloseşte cabluri de cupru scurte şi protejate. Problema sa este că se află în concurenţă cu versiunea cu fibră de înaltă performanţă prezentată mai sus şi cu versiunea ieftină UTP de mai jos. Este destul de puţin probabil să fie folosită la scară largă, în cele din urmă.

Ultima opţiune se referă la smocuri de patru cabluri UTP de categoria 5 lucrând împreună. Deoarece aceste cabluri sunt deja instalate în multe cazuri, este probabil că acest Ethernet gigabit va fi cel adoptat de clienţii cu buzunare strâmte.

Ethernetul gigabit foloseşte reguli noi de codificare pe fibre. Codificarea Manchester la 1 Gbps ar ave anevoie de un semnal de 2 Gbaud, care a fost considerat foarte dificil şi de asemenea foarte risipitor în ceea ce priveşte banda. A fost aleasă în loc o nouă schemă, numită 8B/10B, bazată pe canale de fibră. Fiecare octet de 8 biţi este codificat pe fibră ca 10 biţi, de unde şi denumirea de 8B/10B. Din moment ce există 1024 cuvinte de cod de ieşire pentru fiecare octet de intrare, exista un oarecare spaţiu de alegere în ceea ce priveşte cuvintele care să fie permise. Următoarele două reguli au fost folosite pentru a lua o decizie:

1. Nici un cuvânt de cod nu poate avea mai mult de patru biţi identici la rând;
2. Nici un cuvânt de cod nu poate avea mai mult de şase de 0 sau şase de 1.

Aceste alegeri urmăreau să păstreze destule transmisiuni pe flux pentru a se asigura că receptorul rămâne sincronizat cu emiţătorul, şi de asemenea pentru a păstra numărul de 0-uri şi de l-uri pe fibră pe cât posibil egale între ele. În plus, pentru mulţi octeţi de intrare există două cuvinte de cod care pot fi atribuite. Când codificatorul are de făcut o alegere, va alege întotdeauna varianta care va egaliza numărul de 0 şi 1 transmişi până la momentul respectiv. Accentul este pus pe echilibrarea 0- urilor şi l-urilor pentru a păstra componenta continuă a semnalului la un nivel cât mai scăzut cu putinţă şi pentru a-i permite să treacă nemodificată prin transformatoare. Deşi cercetătorii în domeniul calculatoarelor nu sunt prea încântaţi de faptul că proprietăţile transformatoarelor le dictează schemele de codificare, aşa se întâmplă în viaţă uneori.

Ethernetul gigabit care foloseşte 1000Base-T utilizează o schemă diferită de codificare deoarece sincronizarea datelor pe un cablu de cupru într-un interval de

1ns este prea dificilă. Această soluţie foloseşte patru cabluri torsadate de categorie 5 pentru a permite unui număr de 4 simboluri să fie transmise în paralel. Fiecare simbol este codificat folosind unul din cele cinci niveluri de voltaj. Această schemă permite ca un singur simbol să fie codificat 00, 01, 10, 11 sau cu o valoare specială în scop de control. Prin urmare, există doi biţi de date per pereche torsadată, sau 8 biţi de date per ciclu de ceas. Ceasul funcţionează la 125 MHz, permiţând operarea la 1 Gbps. Motivul pentru care sunt permise cinci niveluri de voltaj în loc de patru este necesitatea de a avea combinaţii rămase disponibile în scopuri de control şi delimitare.

O viteză de 1 Gbps este destul de mare. De exemplu, dacă un receptor este ocupat cu o altă sarcină chiar pentru 1 ms şi nu goleşte spaţiul tampon de pe vreo linie, până atunci este posibil să se fi acumulat chiar şi 1953 cadre, în acel interval de 1 ms. De asemenea, dacă un calculator care foloseşte Ethernet gigabit transmite date unui calculator care foloseşte Ethernet clasic, este foarte probabil ca memoria tampon a celui din urmă să fie epuizată, iar cadrele următoare să fie pierdute. Ca o consecinţă a acestor două observaţii, Ethernetul gigabit suportă fluxuri de control (ca şi Ethernetul rapid, deşi cele două sunt diferite).

Flux de control înseamnă că un capăt trimite un cadru special de control către celălalt capăt, spunându-i să ia o pauză pentru o anumită perioadă de timp. Cadrele de control sunt în general cadre Ethernet având tipul 0x8808. Primii doi octeţi din câmpul de date dau comanda; următorii octeţi oferă parametrii, dacă există vreunul. Pentru fluxul de control sunt folosiţi cadre PAUSE, în care parametrii specifică lungimea pauzei, în unităţi de durată minimă a cadrului. Pentru Ethernetul gigabit unitatea de timp este de 512 ns, permiţând pauze de maxim 33,6 ms.

Imediat după ce Ethernetul gigabit a fost standardizat, comitetul 802 s-a plictisit şi îşi dorea să treacă înapoi la treabă. IEEE le-a spus să înceapă să lucreze la un Ethernet de 10-gigabit. După ce au căutat îndelung o literă care să-i urmeze lui z, au abandonat această abordare şi au trecut la sufixe din două litere. S-au apucat de treabă şi standardul a fost aprobat de IEEE în 2002 ca 802.3ae. Oare cât de departe poate fi Ethernetul de 100-gigabit?

4.3.9 IEEE 802.2: Controlul legăturilor logice

Acum este momentul să ne întoarcem la discuţiile anterioare şi să comparăm ce am învăţat în acest capitol cu ce am studiat în capitolul precedent. În cap. 3 am văzut cum două calculatoare pot comunica sigur printr-o linie nesigură folosind diferite protocoale de legături de date. Aceste protocoale ofereau controlul erorilor (prin mesaje de confirmare) precum şi controlul fluxului de date (folosind o fereastră glisantă).

Dimpotrivă, în acest capitol nu am vorbit deloc despre comunicaţii stabile. Tot ceea ce oferă Ethernetul, ca şi celelalte protocoale 802, este un serviciu datagramă de tipul „best-effort” (cea mai bună încercare). Uneori, acest serviciu este adecvat. De exemplu, în cazul transportării pachetelor IP, nu sunt cerute şi nici măcar nu sunt aşteptate garanţii. Un pachet IP poate să fie inserat într-un câmp de informaţie

utilă 802 şi trimis încotro o fi. Dacă se pierde, asta e.

Totuşi, există şi sisteme în care este de dorit un protocol de legătură de date cu control al erorilor şi al fluxului. IEEE a definit un astfel de protocol care poate funcţiona peste Ethernet şi peste celelalte protocoale 802. Mai mult, acest protocol, numit LLC (Logical Link Control, rom: controlul legăturilor logice), ascunde diferenţele între diferitele tipuri de reţele 802, oferind un singur format şi o singură interfaţă pentru nivelul reţea. Formatul, interfaţa şi protocolul sunt bazate îndeaproape pe protocolul HDLC, pe care l-am studiat în cap. 3. LLC formează jumătatea superioară a nivelului legătură de date, având nivelul MAC dedesubt, după cum se vede în fig. 4-24.

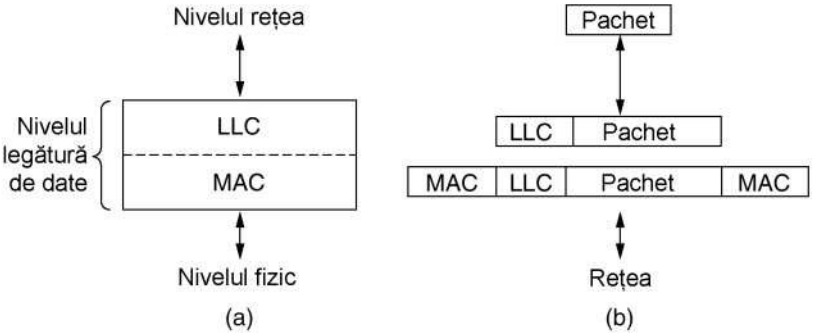


Fig. 4-24. (a) Poziţia LLC. (b) Formatul protocoalelor.

Utilizarea tipică a LLC este prezentată în continuare. Nivelul reţea de pe calculatorul emiţător trimite un pachet către LLC, folosind primitivele de acces LLC. Subnivelul LLC adaugă apoi un antet LLC, conţinând numere care indică secvenţa şi mesajul de confirmare. Structura rezultată este inserată apoi în câmpul de informaţie utilă al unui cadru 802 şi apoi transmisă. Când cadrul ajunge la receptor se desfăşoară procesul invers.

LLC oferă trei opţiuni de servicii: servicii pentru datagrame nesigure, confirmarea serviciului de datagrame, şi un serviciu sigur orientat spre conexiuni. Antetul LLC conţine trei câmpuri: un punct de acces de destinaţie, un punct de acces sursă şi un câmp de control. Punctul de acces spune din partea cărui proces a sosit cadrul şi unde trebuie transportat, înlocuind câmpul „tip” DIX. Câmpul de control conţine numere de secvenţă şi de confirmare, în stilul lui HDLC (vezi fig. 3-24), dar nu identic cu acesta. Aceste câmpuri sunt folosite în principal atunci când este necesară o conexiune stabilă la nivelul legătură de date, caz în care ar fi folosite protocoale similare cu cele discutate în cap. 3. Pentru Internet, încercările de a transmite pachete IP fără garanţii sunt suficiente, astfel încât nu este nevoie de confirmări la nivelul LLC.

4.3.10 Retrospectiva Ethernetului

Ethernetul funcţionează deja de 20 de ani şi încă nu există competitori serioşi, aşa că probabil va mai funcţiona încă mulţi ani. Puţine arhitecturi CPU, sisteme de

operare sau limbaje de programare au dominat scena pentru două sau trei decenii. Evident, Ethernetul a făcut ceva cum trebuie. Ce anume?

Probabil că motivul principal al longevităţii sale este că Ethernetul este simplu şi flexibil. Din punct de vedere practic simplu înseamnă: stabil, ieftin, şi uşor de întreţinut. Odată ce conectorii vampir au fost înlocuiţi de conectori BNC, eşecurile au devenit extrem de rare. Oamenii ezită să înlocuiască ceva ce merge perfect tot timpul, mai ales când ştiu că o grămadă de lucruri din industria calculatoarelor merg foarte prost, astfel încât multe dintre aşa numitele îmbunătăţiri funcţionează semnificativ mai prost decât versiunea pe care au înlocuit-o.

Simplu înseamnă de asemenea ieftin. Ethernetul subţire şi cablarea cu cabluri torsadate sunt relativ ieftine. Plăcile de reţea nu sunt nici ele scumpe. Doar când au fost introduse concentratoarele şi comutatoarele au fost necesare investiţii substanţiale, dar în momentul în care acestea au apărut în peisaj, Ethernetul era deja solid stabilit.

Ethernetul este uşor de întreţinut. Nu trebuie instalat nici un software (în afara driver-elor) şi nu există tabele de configuraţii care să trebuiască administrate (şi astfel să fie un prilej de greşeli). De asemenea, adăugarea unor noi staţii nu înseamnă nimic mai mult decât introducerea unui cablu în placa lor de reţea.

Un alt aspect este faptul că Ethernetul se integrează uşor cu TCP/IP, care a devenit dominant. IP este un protocol fără conexiune, ceea ce se potriveşte perfect cu Ethernetul, care nici el nu este orientat pe conexiune. De exemplu, IP se potriveşte mult mai greu cu ATM, care este orientat spre conexiune şi această nepotrivire este un dezavantaj serios în impunerea ATM.

În cele din urmă, Ethernetul a fost capabil să evolueze în anumite aspecte cruciale. Vitezele au crescut cu câteva ordine de mărime, au fost introduse concentratoarele şi comutatoarele, iar aceste schimbări nu au necesitat schimbarea interfeţelor software. Dacă un vânzător din domeniul reţelelor vă arată o instalaţie amplă şi vă spune „am această nouă reţea fantastică pentru Dvs. Tot ce trebuie să faceţi este să vă aruncaţi tot hardware-ul şi să vă rescrieţi tot software-ul”, atunci are o problemă. FDDI, Canal de fibră şi ATM au fost toate mai rapide decât Ethernetul când au fost introduse, dar erau incompatibile cu Ethernetul, mult mai complexe şi mai dificil de administrat. În cele din urmă Ethernetul le-a ajuns din urmă în ceea ce priveşte viteza, astfel încât, rămase fără nici un avantaj, au murit în tăcere – cu excepţia ATM care este folosit în interiorul sistemului de telefonie.

4.4 REŢELE LOCALE FĂRĂ FIR

Deşi Ethernetul este folosit pe scară largă, competiţia este pe cale să apară. LAN-urile fără fir sunt din ce în ce mai populare, şi tot mai multe clădiri, aeroporturi şi alte spaţii publice sunt echipate cu ele. LAN-urile fără fir pot opera în două configuraţii, după cum am văzut în fig. l-35: cu sau fără staţie de bază. Prin urmare, standardul LAN 802.11 ia acest fapt în considerare şi oferă sprijin pentru

ambele aranjamente, după cum vom vedea în continuare.

Am oferit nişte informaţii introductive despre 802.11 în secţiunea 1.5.4. Acum este momentul să ne uităm mai îndeaproape la tehnologie. În secţiunile următoare ne vom uita la stiva de protocoale, la tehnicile de la nivelul fizic radio de transmisiuni, la protocolul subnivelului MAC, la structura cadrelor şi la servicii. Pentru mai multă informaţie despre 802.11 vezi (Crov et. al., 1997; Geier, 2002; Heegard et. al, 2001; Kapp, 2002; O'Hara şi Petrick, 1999; Severance, 1999). Pentru a afla adevărul chiar de la sursă, consultaţi standardul publicat al 802.11.

4.4.1. Stiva de protocoale 802.11

Protocoalele folosite de toate variantele 802, inclusiv Ethernetul, au o anumită similaritate a structurii. O viziune parţială a stivei de protocoale 802.11 este prezentată în fig. 4-25. Nivelul fizic
corespunde destul de bine cu nivelul fizic OSI, dar nivelul de legătură de date în toate protocoalele 802 este divizat în două sau mai multe subniveluri. În 802.11, subnivelul MAC (Medium Access Control) determina alocarea canalului, şi anume cine va transmite următorul. Deasupra sa se află subnivelul LLC (Logical Link Control), a cărui treabă este să ascundă diferenţele dintre diferitele variante 802 şi să le facă să pară la fel pentru nivelul reţea. Am studiat LLC mai devreme în acest capitol, atunci când am discutat Ethernetul, astfel încât nu vom repeta această informaţie aici.

Standardul 802.11 din 1997 specifică trei tehnici de transmisie permise la nivelul fizic. Metoda în- fraroşu foloseşte cam aceeaşi tehnologie ca şi telecomenzile TV. Celelalte două folosesc transmisia radio pe distanţă scurtă, prin tehnici denumite FHSS şi DSSS. Ambele utilizează o parte a spectrului care nu necesită licenţe (banda 2,4 GHz ISM). Uşile de garaj care se deschid prin mesaje radio folosesc tot această parte a spectrului, astfel încât calculatorul vostru s-ar putea afla în competiţie cu uşa de la garaj. Telefoanele fără fie şi cuptoarele cu microunde folosesc şi ele această bandă. Toate aceste tehnici operează la 1Mbps sau 2Mbps şi cu putere suficient de mică astfel încât nu intră prea mult în conflict. În 1999 au fost introduse două noi tehnici pentru a obţine o bandă mai largă. Acestea sunt denumite OFDM şi HR-DSSS. Ele operează până la 54Mbps şi respectiv 11Mbps. În 2001, o a doua modulaţie OFDM a fost introdusă, dar într-o bandă de frecvenţă diferită de prima. În continuare le vom examina pe fiecare pe scurt. Din punct de vedere tehnic acestea aparţin nivelului fizic şi ar fi trebuit să fie examinate în cap. 2, dar deoarece sunt atât de strâns legate de reţelele locale în general şi de subnivelul 802.11 MAC, le abordăm mai degrabă aici.

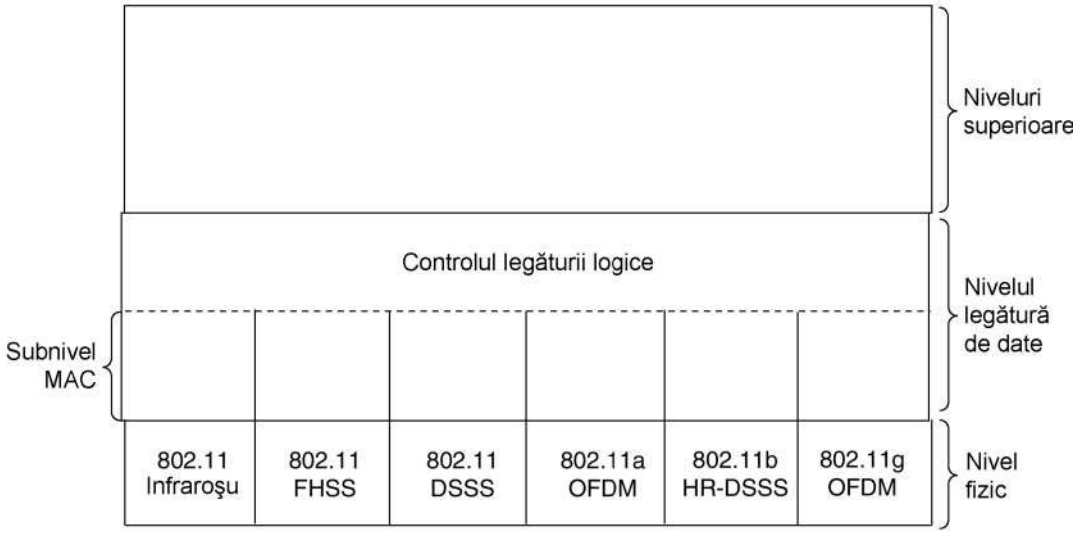


Fig. 4-25. Parte a stivei protocolului 802.11.

4.4.2. Nivelul fizic al 802.11

Fiecare dintre cele cinci tehnici de transmitere permise face posibilă trimiterea unui cadru MAC de la o staţie la alta. Totuşi, ele diferă în ceea ce priveşte tehnologia folosită şi vitezele la care pot ajunge. O discuţie detaliată a acestor tehnologii este cu mult în afara cadrului acestei cărţi, dar câteva cuvinte despre fiecare, precum şi menţionarea cuvintelor cheie vor oferi cititorilor interesaţi termenii necesari pentru a găsi mai multă informaţie pe Internet sau în altă parte.

Soluţia bazată pe infraroşu foloseşte transmisiuni cu difuzare (adică fără vizibilitate directă) la 0,85 sau 0,95 microni. Sunt permise două viteze: IMbps şi 2Mbps. La IMbps se foloseşte o schemă de codificare în care un grup de 4 biţi este codificat ca un cuvânt de 16 biţi conţinând 15 de 0 şi un singur 1, prin ceea ce se numeşte **codul Gray**. Acest cod are proprietatea că o mică eroare în sincronizarea temporală duce doar la o eroare de un bit în output. La 2Mbps, codificarea ia 2 biţi şi produce un cuvânt codificat de 4 biţi, de asemenea cu un singur 1 – adică unul dintre 0001, 0010, 0100 şi 1000. Semnalele infraroşii nu pot trece prin ziduri, deci celulele din camere diferite sunt bine izolate unele de altele. Totuşi, datorită lărgimii de bandă reduse (şi faptului că lumina soarelui afectează semnalele în infraroşu), aceasta nu este o opţiune populară.

FHSS (**Frequency Hopping Spread Spectrum**, rom: salturi de frecvenţă într-un spectru larg) foloseşte 79 de canale, fiecare de 1MHz, începând la nivelul inferior al benzii de 2.4GHz ISM. Un generator de numere pseudo-aleator este utilizat pentru a produce secvenţa de frecvenţe după care se vor efectua salturile. Cât timp staţiile folosesc aceeaşi rădăcină pentru generatorul de numere pseu- do-aleatoare şi stau sincronizate, ele vor sări simultan la aceleaşi frecvenţe. Durata petrecută pe fiecare frecvenţă, denumită „timpul de locuire”, este un parametru ajustabil, dar trebuie să fie mai mică de 400 ms. Factorul aleator al FHSS oferă o metodă eficientă de alocare a spectrului în banda ISM care nu este reglementată. De asemenea, oferă

un minimum de securitate, deoarece un intrus care nu cunoaşte secvenţa de salt sau timpul de locuire nu poate trage cu urechea la transmisiuni. Dacă distanţele sunt mari, apare problema atenuării la transmisia pe mai multe căi, dar FHSS oferă o rezistenţă bună. Este de asemenea relativ insensibil la interferenţa radio, ceea ce îl face popular pentru legăturile dintre clădiri. Principalul său dezavantaj îl constituie lărgimea de bandă redusă.

Ce-a de-a treia metodă de modulare, **DSSS** (**Direct Sequence Spread Spectrum**, rom: spectru larg cu succesiune directă) este de asemenea limitată la 1Mbps sau 2Mbps. Această schemă are anumite similarităţi cu sistemul CDMA pe care l-am examinat în secţiunea 2.6.2, dar diferă în alte privinţe. Fiecare bit este transmis ca o secvenţă de 11 fragmente, folosind ceea ce se numeşte **secvenţă Barker**. Este folosită schimbarea modulării în fază la 1Mbaud, transmiţând un bit per baud când operează la 1Mbps şi 2 biţi per baud când operează la 2Mbps. O bună bucată de timp, FCC a cerut ca toate echipamentele de comunicaţii fără fir care operează în banda ISM în Statele Unite să folosească împrăştierea spectrului, dar această regulă a fost abandonată în mai 2002, datorită apariţiei unor noi tehnologii.

Prima dintre reţelele locale fără fir de mare viteză, 802.11a, foloseşte **OFDM** (**Orthogonal Frequency Division Multiplexing**, rom: multiplexare cu divizare în frecvenţe ortogonale) pentru a transmite până la 54Mbps în banda mai largă de 5GHz ISM. După cum sugerează şi termenul FDM, sunt folosite diferite frecvenţe – un număr de 52 de frecvenţe, 48 pentru date şi 4 pentru sincronizare – similar cu ADSL. Din moment ce transmisiunile sunt prezente pe frecvenţe multiple în acelaşi timp, această tehnică este considerată o formă de împrăştiere a spectrului, dar diferită de CDMA şi FHSS. Divizarea semnalului în mai multe benzi înguste are anumite avantaje comparativ cu folosirea unei benzi unice largi, incluzând o imunitate mai bună la interferenţa de bandă îngustă şi posibilitatea utilizării benzilor care nu sunt contigue. Este folosit un sistem complex de codificare, bazat pe modularea schimbării de fază pentru viteze de până la 10 Mbps şi pe QAM la viteze superioare. La 54 Mbps, 216 de biţi de date sunt codificaţi în simboluri de 288 de biţi. O parte din motivaţia pentru OFDM este compatibilitatea cu sistemul European HiperLAN/2 (Doufexi et al., 2002). Tehnica are o eficienţă bună a spectrului în termeni de biţi / Hz şi o imunitate bună în faţa atenuării la transmisia pe mai multe căi.

În continuare, am ajuns la **HR-DSSS** (**High Rate Direct Sequence Spread Spectrum**, rom: spectru larg cu succesiune directă la rată ridicată), o altă tehnică de spectru larg, care foloseşte 11 milioane de fragmente/secundă pentru a obţine 11Mbps în banda de 2,4 GHz. Este denumită **802.11b**, dar nu este o continuare pentru 802.11a. De fapt, standardul său a fost aprobat primul şi lansat pe piaţă primul. Vitezele de date suportate de 802.11b sunt 1, 2, 5,5 şi 11 Mbps. Cele două viteze reduse se obţin la 1Mbaud, cu 1 şi respectiv 2 biţi per baud, folosind modularea schimbării de fază (pentru a fi compatibil cu DSSS). Cele două viteze mai mari se obţin la 1,375 Mbaud, cu 4 şi respectiv 8 biţi per baud, folosind codurile **Walsh/Haramard**. Viteza datelor poate fi adaptată dinamic în timpul operării, pentru a obţine viteza optimă posibilă în condiţiile curente de încărcare şi

zgomot. În practică, viteza operaţională a lui 802.11b este de aproape 11Mbps. Deşi 802.11b este mai încet decât 802.11a, spaţiul său de variaţie este de aproape şapte ori mai mare, ceea ce este mai important în multe situaţii.

O versiune îmbunătăţită a 802.11b, 802.11g, a fost aprobată de IEEE în noiembrie 2001, după multe dispute în legătură cu ce tehnologie patentată să folosească. Aceasta utilizează modulaţia OFDM a versiunii 802.11a, dar operează în banda îngustă 2,4 GHz ISM ca şi 802.11b. Teoretic vorbind, poate opera la peste 54Mbps. Nu este încă foarte clar precizat momentul când această viteză va fi realizată în practică. Prin urmare, comitetul 802.11 a produs trei reţele locale fără fir: 802.11a, 802.11b şi 802.11c (fără să menţionăm trei reţele locale fără fir de viteză redusă). Ne-am putea pune întrebarea legitimă dacă asta este un lucru bun pentru un comitet de standardizare. Poate că trei este numărul lor cu noroc.

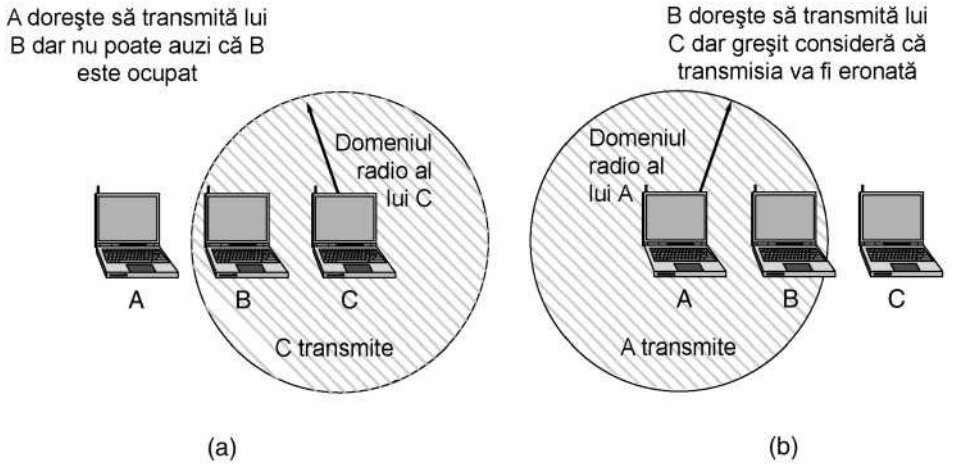


Fig. 4-26. (a) Problema staţiei ascunse. (b) Problema staţiei expuse.

4.4.3 Protocolul subnivelului MAC al 802.11

Să ne întoarcem acum din domeniul ingineriei electrice în domeniul ştiinţei calculatoarelor. Protocolul subnivelului MAC al 802.11 este destul de diferit de acela al Ethernetului datorită complexităţii inerente a mediului fără fir, comparativ cu un sistem de cabluri. În Ethernet o staţie aşteaptă până când eterul a tăcut şi apoi începe să transmită. Dacă nu aude un zgomot de ciocnire în primii 64 de octeţi înseamnă ca aproape sigur cadrul a fost recepţionat corect. În cazul reţelelor fără fir, situaţia e diferită.

Pentru început, apare problema staţiei ascunse menţionată anterior şi ilustrată în fig. 4-26(a). Din moment ce nu toate staţiile se află în domeniul de acces radio una faţă de alta, transmisiunile care se petrec într-o parte a celulei pot să nu fie recepţionate în altă parte a celulei. În acest exemplu, staţia C transmite staţie B. Dacă A observă canalul, nu va auzi nimic şi va trage concluzia falsă că poate acum să înceapă să-i transmită lui B.

În plus, există şi problema inversă, a staţiei expuse, ilustrată în fig. 4-26(b). Aici

B vrea să-i trimită lui C astfel încât observă canalul. Când aude o transmisiune trage concluzia falsă că nu poate să-i transmită lui C, chiar dacă de fapt A îi transmite lui D (care este absent din imagine).

Mai mult, majoritatea radiourilor sunt semi-duplex, ceea ce înseamnă că nu pot transmite şi asculta zgomotele de coliziuni simultan pe aceeaşi frecvenţă. Ca urmare a acestor probleme, 802.11 nu utilizează CSMA/CD utilizat de Ethernet.

Pentru a face faţă acestor probleme, 802.11 suportă două tipuri de operaţii. Primul, denumit **DCF** (**Distributed Coordination Function**, rom: funcţie de coordonare distribuită), nu foloseşte nici un fel de control central (fiind similar în această privinţă Ethernetului). Celălalt, denumit **PCF** (**Point Coordination Function**, rom: funcţie de coordonare punctuală), foloseşte staţia de bază pentru a coordona toată activitatea din celula sa. Toate implementările trebuie să poată susţine DCF, dar PCF este opţional. Vom discuta acum aceste două modalităţi, alternativ.

Când este folosit DCF, 802.11 utilizează un protocol denumit **CSMA/CA** (**CSMA with Collision Avoidance**, rom: CSMA cu evitarea coliziunilor). În acest protocol se foloseşte atât observarea canalelor fizice, cât şi observarea canalelor virtuale. CSMA/CA suportă două tipuri de operaţii. În prima metodă, atunci când o staţie vrea să transmită, observă canalul. Dacă este liber, începe să transmită. Nu va mai asculta canalul în timpul transmisiunii, ci va trimite întregul cadru, care ar putea foarte bine să fie distrus la receptor datorită interferenţei. Dacă în schimb canalul este ocupat, emiţătorul amână transmisia până când mediul se eliberează şi abia apoi începe să transmită. Dacă apare o coliziune, staţiile implicate aşteaptă un timp aleatoriu, folosind algoritmul exponenţial de regresie binară, şi mai încearcă o dată.

Cealaltă metodă a CSMA/CA se bazează pe MACAW şi foloseşte observarea canalelor virtuale, după cum este ilustrat în fig. 4-27. În acest exemplu, *A* vrea să transmită către *B*. *C* este o staţie din sfera de recepţie a lui *A* (posibil şi a lui *B*, dar nu contează). *D* este o staţie din sfera lui *B* dar în afara sferei lui *A*.

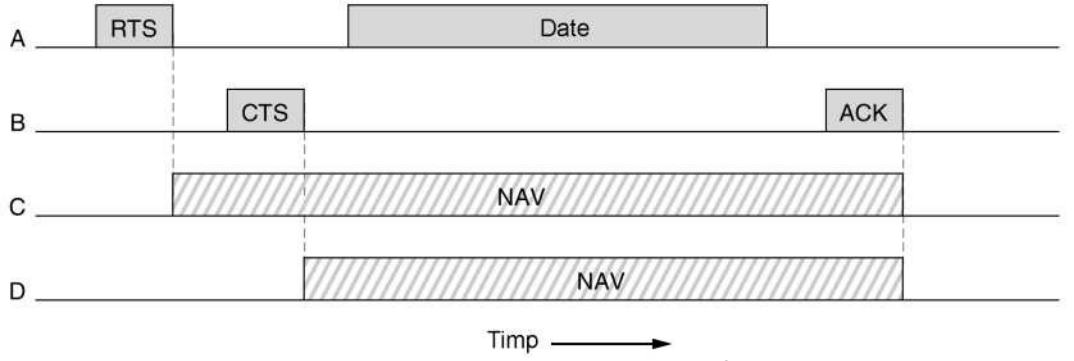


Fig. 4-27. Utilizarea canalului virtual utilizând CSMA/CA.

Protocolul începe când *A* decide că vrea să emită către *B*. Începe prin a trimite un cadru RTS către *B*, pentru a-i cere permisiunea să îi trimită un cadru. Când *B* primeşte această cerere, se poate decide să ofere permisiunea, caz în care trimite

un cadru CTS înapoi. După ce primeşte CTS, *A* îşi trimite cadrul şi iniţiază un cronometru pentru ACK. După recepţionarea corectă a cadrului de date, *B* răspunde cu un cadru ACK, terminând schimbul. În cazul în care cronometrul pentru ACK a lui *A* expiră înainte ca ACK să revină la el, întregul protocol este luat de la capăt.

Să ne uităm acum la acest schimb din perspectiva lui *C* şi a lui *D*. *C* este în sfera lui *A*, deci poate primi cadrul RTS. Dacă îl primeşte, îşi va da seama că o staţie urmează să emită date în curând, astfel încât se va abţine să transmită orice până când schimbul este complet. Din informaţia prezentă în cererea RTS poate estima durata tranzacţiei, incluzând ACK-ul final, astfel încât îşi simulează o ocupare virtuală, indicată în fig. 4-27 de **NAV** (**Network Allocation Vector**, rom: vector de alocare a reţelei). *D* nu aude RTS, dar aude CTS, astfel încât de asemenea îşi alocă un NAV. Observaţi că semnalele NAV nu sunt transmise; ele sunt doar modalităţi interne de a aminti staţiilor să tacă pentru o anumită perioadă.

Spre deosebire de reţelele cablate, reţelele fără fir sunt zgomotoase şi instabile, într-o oarecare măsură şi datorită cuptoarelor cu microunde, care folosesc şi ele banda fără licenţă ISM. Ca urmare, probabilitatea unui cadru de a ajunge la destinaţie cu succes scade odată cu creşterea lungimii cadrului. Dacă probabilitatea unei erori la nivel de 1 bit este p, atunci probabilitatea unui cadru de n biţi de a ajunge corect la final este $(1-p)^n$. De exemplu, dacă $p=10^{-4}$, probabilitatea de a recepţiona corect un cadru Ethernet întreg (12.144 biţi) este mai mică de 30%. Dacă $p=10^{-5}$, aproximativ un cadru din 9 va fi afectat. Chiar dacă $p=10^{-6}$, peste 1% dintre cadre vor fi afectate, ceea ce înseamnă aproape o duzină pe secundă, şi chiar mai multe dacă se folosesc cadre mai scurte decât valoarea maximă. Pe scurt, dacă un cadru este prea lung, are puţine şanse să fie transmis fără avarii, şi va trebui, cel mai probabil, retransmis.

Pentru a rezolva problema canalelor zgomotoase, 802.11 permite cadrelor să fie fragmentate în bucăţi mai mici, fiecare cu propria sumă de control. Fragmentele sunt numerotate individual şi confirmate folosind un protocol pas-cu-pas (emiţătorul nu poate transmite fragmentul k+1 decât după ce a primit confirmarea pentru fragmentul k). Odată ce canalul a fost obţinut prin RTS şi CTS, mai multe fragmente pot fi trimise într-un şir, ca în fig. 4-28, secvenţa purtând numele de **rafală de fragmente** (**fragment burst**).

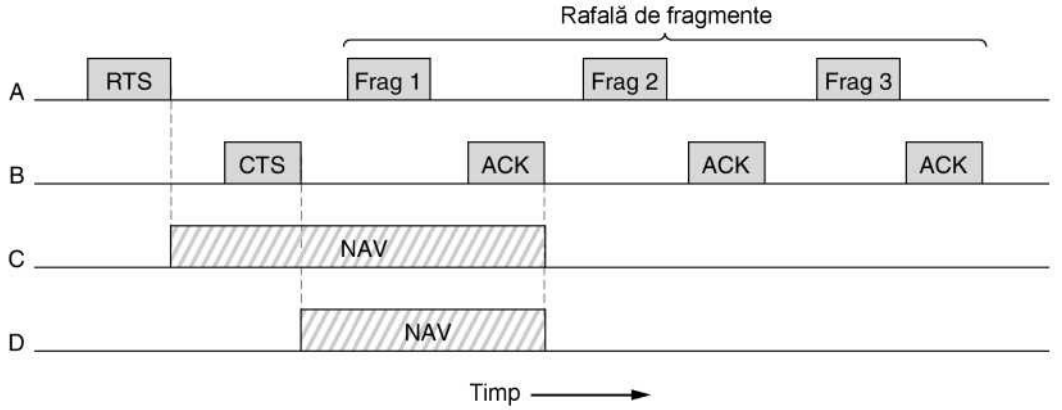


Fig. 4-28. O rafală de fragmente.

Fragmentarea creşte productivitatea prin restricţionarea retransmiterii doar la fragmentele eronate, eliminând necesitatea de a retransmite întregul cadru. Dimensiunea fragmentului nu este fixată de standard, ci este un parametru pentru fiecare celulă şi poate fi ajustată de către staţia de bază. Mecanismul NAV menţine celelalte staţii tăcute numai până la confirmarea următoare, dar un alt

mecanism (descris în cele ce urmează) este utilizat pentru a permite unei întregi rafale de fragmente să fie transmisă fără interferenţe.

Toată discuţia anterioară se aplică numai modului DCF 802.11. În acest mod, nu există control centralizat, iar staţiile sunt în competiţie pentru timpul de transmisie, exact ca la Ethernet. Celălalt mod permis este PCF, în care staţia de bază interoghează celelalte staţii întrebându-le dacă au cadre de transmis. Din moment ce, în mod PCF, ordinea transmisiilor este complet controlată de către staţia de bază, nu apar niciodată coliziuni. Standardul prescrie mecanismul pentru interogare, dar nu frecvenţa interogărilor, ordinea interogărilor şi nici măcar dacă staţiile trebuie să primească toate drepturi egale.

Mecanismul principal este acela prin care staţia de bază emite periodic (de 10 până la 100 de ori pe secundă) un **cadru baliză** (**beacon frame**). Cadrul baliză conţine parametrii de sistem, cum ar fi intervalul de salt şi timpii de viaţă (pentru FHSS), sincronizări de ceas, etc. De asemenea, acest cadru invită staţiile noi să se înregistreze pentru serviciul de interogare. Din momentul în care o staţie s-a înregistrat pentru serviciul de interogare la o anumită viteză, aceasta va beneficia garantat de o fracţiune din lăţimea de bandă, în acest fel fiind posibilă oferirea de garanţii de tip calitatea-serviciului.

Viaţa bateriei este întotdeauna o problemă în cazul dispozitivelor mobile fără fir, deci 802.11 acordă atenţie problemei gestionării consumului. În particular, staţia de bază poate instrui un dispozitiv mobil să intre în aşteptare până când este trezit în mod explicit de către staţia de bază sau de către utilizator. Totuşi, punerea unei staţii în aşteptare presupune ca staţia de bază să aibă responsabilitatea stocării în zone tampon a cadrelor direcţionate către staţia respectivă atâta timp cât aceasta este în aşteptare. Aceste cadre vor putea fi colectate ulterior.

Modurile PCF şi DCF pot coexista în cadrul aceleiaşi celule. La prima vedere, pare imposibilă prezenţa unei scheme de control centralizate şi a unei scheme distribuite în acelaşi timp, dar 802.11 oferă o modalitate de a atinge acest scop. Modul de funcţionare este atins prin definirea atentă a intervalului de timp dintre transmisia de cadre. După ce un cadru a fost transmis, un anumit interval de timp este impus staţiei înainte de a putea transmite următorul cadru. Sunt definite patru intervale diferite, fiecare pentru scopuri precise. Cele patru intervale sunt prezentate în fig. 4-29.

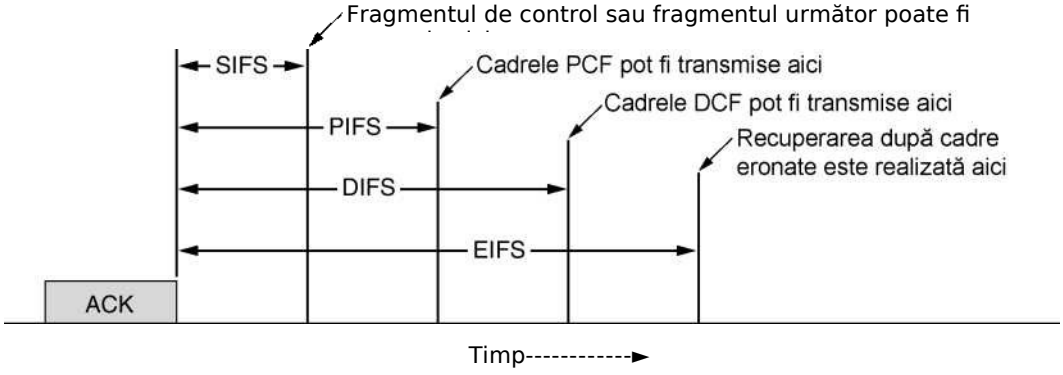


Fig. 4-29. Spaţierea între cadre în

Cel mai scurt interval este **SIFS** (**Short InterFrame Spacing**, rom: spaţiere redusă

între cadre). Este folosit pentru a permite părţilor implicate într-un dialog singular să transmită primele. Această soluţie oferă posibilitatea receptorului să trimită un CTS ca răspuns la un RTS, acordă permisiunea ca un receptor să trimită o confirmare pentru un fragment sau pentru un cadru integral şi acordă
permisiunea ca emiţătorul unui fragment să trimită în rafală următorul fragment fără a fi nevoie să aştepte din nou un RTS.

Întotdeauna există exact o singură staţie care are dreptul să răspundă după un interval SIFS. Dacă aceasta nu reuşeşte să profite de această oportunitate şi se scurge un interval de timp **PIFS** (**PCF InterFram Interval**, rom: interval între cadrele PCF), staţia de bază poate transmite un cadru baliză sau un cadru de interogare. Acest lucru permite unei staţii care transmite un cadru de date sau o secvenţă de fragmente să termine cadrul fără să se interpună nimeni, dar în acelaşi timp dă staţiei de bază o şansă să acapareze canalul atunci când emiţătorul anterior a terminat, fără să trebuiască să intre în competiţie cu utilizatorii nerăbdători.

Dacă staţia de bază nu are nimic de transmis şi se scurge un interval de timp **DIFS (DCF InterFrame Interval**, rom: interval între cadrele DCF), orice staţie poate încerca obţinerea unui canal pentru a transmite un nou cadru. Regulile obişnuite de competiţie se aplică şi o reluare după un timp binar exponenţial poate fi necesară în cazul apariţiei unei coliziuni.

Ultimul interval de timp, **EIFS (Extended InterFrame Spacing**, rom: spaţiere extinsă între cadre), este utilizat pentru a raporta eventuale erori de către o staţie care tocmai a primit un cadru eronat sau necunoscut. Ideea este ca acest eveniment să aib ă cea mai mică prioritate, pentru că, din moment ce receptorul s-ar putea să nu ştie ce se întâmplă, acesta ar trebui să aştepte un interval substanţial de timp pentru a evita interferenţa cu un dialog în desfăşurare între două staţii.

4.4.4 Formatul cadrului 802.11

Standardul 802.11 defineşte trei clase diferite de cadre: de date, de control şi de gestionare. Fiecare dintre acestea are un antet cu o varietate de câmpuri folosite în cadrul subnivelului MAC. În plus, există unele antete utilizate de către nivelul fizic, dar acestea sunt de obicei legate de modalităţile de modulaţie folosite, deci nu le vom discuta aici.

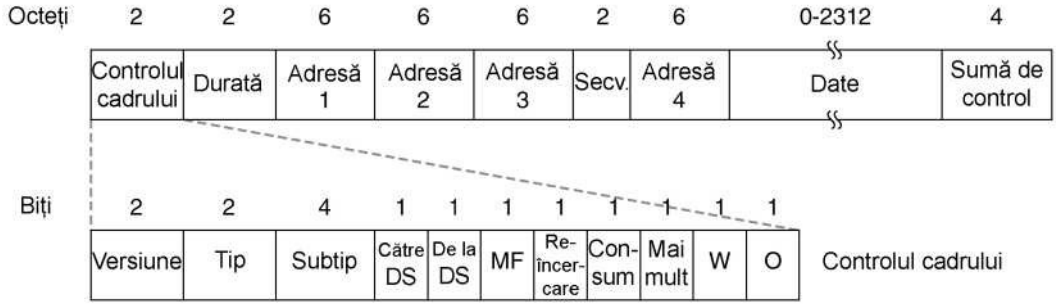


Fig. 4-30. Cadrul de date 802.11.

Formatul cadrului de date este prezentat în fig. 4-30. Primul câmp este acela de *control al cadrului.* Acesta are la rândul lui 11 subcâmpuri. Primul dintre acestea este *versiunea de protocol,* care permite celor două versiuni ale protocolului să opereze în acelaşi timp în aceiaşi celulă. Apoi urmează câmpurile pentru *tip* (de date, de control sau de gestiune) şi *subtip* (de ex. RTS sau CTS). Biţii *către DS* şi *de la DS* indică direcţia de transport a cadrului – către sistemul de distribuţie sau de la sistemul de distribuţie între celule (de ex. Ethernet). Bitul *MF* semnalizează că vor urma mai multe fragmente. Bitul *reîncercare* marchează o retransmisie a unui cadru trimis anterior. Bitul *gestiunea consumului* este folosit de către staţia de bază pentru a pune receptorul în stare de aşteptare sau pentru a-l scoate din starea de aşteptare. Bitul *mai mult* indică faptul că emiţătorul mai are cadre adiţionale pentru receptor. Bitul *W* specifică criptarea cadrului folosind algoritmul **WEP** (**Wired Equivalent Privacy**, rom. confidenţialitate echivalentă cablată). În sfârşit, bitul *O* indică receptorului că o secvenţă de cadre cu acest bit setat trebuie prelucrată strict în ordinea în care cadrele au fost recepţionate.

Cel de-al doilea câmp al cadrului de date, câmpul *durată,* indică intervalul de timp în care cadrul şi confirmarea vor ocupa canalul. Acest câmp este de asemenea prezent în cadrele de control şi reprezintă modalitatea prin care alte staţii gestionează mecanismul NAV. Antetul cadrului conţine patru adrese, toate în formatul standard IEEE 802. În mod evident, sursa şi destinaţia sunt necesare, dar pentru ce sunt necesare celelalte două? Să ne aducem aminte: cadrele pot intra sau ieşi dintr-o celulă prin staţia de bază. Celelalte două adrese sunt folosite pentru staţiile de bază sursă şi destinaţie în traficul între celule.

Câmpul *secvenţă* permite numerotarea fragmentelor. Din cei 16 biţi disponibili, 12 identifică cadrul şi 4 identifică fragmentul. Câmpul *date* conţine încărcătura utilă, având până la 2312 octeţi, fiind urmat de uzuala *sumă de control.*

Cadrele de gestiune au un format similar celor de date, cu excepţia uneia dintre adresele staţiilor de bază, deoarece cadrele de gestiune sunt restricţionate la o singură celulă. Cadrele de control sunt şi mai scurte, având numai una sau două adrese, neavând câmp de *date* şi nici câmp *secvenţă*. Aici informaţia cheie se află în câmpul *subtip,* de obicei RTS, CTS, sau ACK.

4.4.5 Servicii

Standardul 802.11 afirmă că fiecare reţea locală fără fir trebuie să ofere nouă servicii. Aceste servicii sunt împărţite în două categorii: cinci servicii legate de distribuţie şi patru servicii pentru staţii. Serviciile de distribuţie sunt legate de gestiunea apartenenţei la celulă şi de interacţiunea cu staţiile din afara celulei. Serviciile de staţie sunt legate de activitatea în cadrul unei singure celule.

Cele cinci servicii de distribuţie sunt oferite de către staţiile de bază şi se ocupă de mobilitatea staţiilor pe măsură ce acestea intră şi ies din celule, ataşându-se şi detaşându-se de la staţia de bază. Ele sunt următoarele:

1. **Asocierea**. Acest serviciu este folosit de către staţiile mobile pentru conectare la staţia de bază. De obicei, el este utilizat exact după ce o staţie se deplasează în acoperirea radio a staţiei de bază. La sosire, aceasta îşi anunţă identitatea şi

capacităţile. Capacităţile includ: vitezele de date suportate, cererile de servicii PCF (de exemplu interogarea) şi necesităţile de gestionare a consumului. Staţia de bază poate accepta sau rejecta staţia mobilă. Dacă staţia mobilă este acceptată, atunci aceasta trebuie să se autentifice.

2. **Dezasocierea**. Atât staţia, cât şi staţia de bază se pot dezasocia, rupând în acest fel relaţia. O staţie trebuie să folosească acest serviciu înainte de a se închide sau de a pleca, iar staţia de bază îl poate folosi şi ea – de exemplu înainte de oprirea pentru întreţinere.
3. **Reasocierea**. O staţie îşi poate schimba staţia de bază preferată utilizând acest serviciu. Facilitatea este utilă pentru staţiile mobile care se deplasează dintr-o celulă în alta. Dacă serviciul este folosit corect, nu se vor pierde date la trecere. (Dar 802.11, ca şi Ethernetul, este numai un serviciu fără confirmare).
4. **Distribuţia**. Acest serviciu determină modul în care sunt rutate cadrele trimise către staţia de bază. Dacă destinaţia este locală staţiei de bază, cadrele pot fi trimise direct în aer. În caz contrar, ele vor trebui înaintate prin reţeaua cablată.

5. **Integrarea**. Dacă un cadru trebuie să circule printr-o reţea care nu este 802.11 şi utilizează o schemă de adresare diferită şi un format de cadre diferit, acest serviciu efectuează translatarea de la formatul 802.11 la formatul reţelei destinaţie.

Cele patru servicii rămase sunt servicii în interiorul celulei (adică sunt legate de acţiuni în interiorul unei singure celule). Ele sunt folosite după ce a avut loc asocierea şi sunt următoarele:

1. **Autentificarea.** Deoarece comunicaţiile fără fir pot fi recepţionate sau emise cu uşurinţă de către staţii neautorizate, o staţie trebuie să se autentifice înainte de a i se permite să trimită date. După ce o staţie mobilă a fost asociată de către o staţie de bază (adică acceptată în cadrul celulei), staţia de bază îi va trimite un cadru special de provocare pentru a verifica dacă staţia mobilă cunoaşte cheia secretă (parola) care i-a fost alocată. Staţia va dovedi cunoaşterea cheii prin criptarea cadrului provocare cu ea, urmată de trimiterea acestuia înapoi la staţia de bază. Dacă rezultatul este corect, staţia mobilă este pe deplin înscrisă în celulă. În standardul iniţial, staţia de bază nu trebuia să-şi demonstreze identitatea către staţiile mobile, dar lucrul pentru eliminarea acestui defect din standard este în desfăşurare.
2. **Deautentificarea.** Când o staţie anterior autentificată doreşte să părăsească reţeaua, aceasta este deautentificată. După deautentificare, staţia nu va mai putea folosi reţeaua.
3. **Confidenţialitatea.** Pentru ca informaţiile transmise printr-o reţea fără fir să rămână confidenţiale, acestea trebuie criptate. Acest serviciu gestionează criptarea şi decriptarea. Algoritmul de criptare specificat este RC4, inventat de către Ronald Rivest de la M.I.T.

4. **Livrarea datelor.** În fine, transmisia datelor este subiectul principal, deci în mod evident

802.11 oferă o modalitate de a transmite şi recepţiona date. Din moment ce

802.11 este modelat după Ethernet şi transmisia prin Ethernet nu este 100% sigură, transmisiile prin 802.11 nu sunt garantate nici ele să fie sigure. Nivelurile superioare trebuie detecteze şi să corecteze erorile.

O celulă 802.11 are unii parametrii care pot fi inspectaţi şi, în unele cazuri, ajustaţi. Aceştia sunt legaţi de criptare, intervale de expirare, viteze de transfer pentru date, frecvenţa balizelor şi aşa mai departe.

Reţelele locale fără fir bazate pe 802.11 încep să fie folosite peste tot în lume în clădiri de birouri, hoteluri, restaurante şi campusuri. Se preconizează o creştere rapidă. Pentru unele date legate de desfăşurarea extinsă a 802.11 la CMU, vezi (Hills, 2001).

4.5 REŢELE FĂRĂ FIR DE BANDĂ LARGĂ

Am stat în clădiri prea mult. Haideţi să ieşim afară şi să vedem dacă există reţele interesante acolo. Se pare că se întâmplă destul de multe pe acolo şi o parte dintre acestea au de-a face cu aşa numită ultimă-porţiune (ultima sută de metri). Odată cu deregularizarea serviciilor telefonice în multe ţări, competitori ai companiilor telefonice fortificate au din ce în ce mai des permisiunea să ofere servicii de voce şi de Internet la mare viteză. Există în mod sigur o mare cerere în domeniu. Problema este că desfăşurarea de fibră, de cablu coaxial sau chiar de perechi torsadate de categoria 5 este prohibitiv de scumpă. Ce poate face atunci un competitor?

Răspunsul este simplu: reţele fără fir de bandă largă. Ridicarea unei antene mari pe un deal imediat în afara oraşului şi instalarea de antene direcţionate către ea pe acoperişurile clienţilor este mult mai simplu şi mai ieftin decât săparea de şanţuri şi tragerea de cabluri prin acestea. Prin urmare, companiile de telecomunicaţii concurente au un interes important în a oferi servicii de comunicaţie fără fir la viteze de mulţi megabiţi pe secundă pentru voce, Internet, filme la cerere etc. După cum am văzut în fig. 2-30, LMDS a fost inventat pentru acest scop. Totuşi, până de curând, fiecare companie şi-a dezvoltat propriul sistem. Această lipsă de standarde a însemnat că software-ul şi har- dware-ul nu au putut fi produse pe scară largă, ceea ce a menţinut preţurile la un nivel mare şi ac- ceptabilitatea la un nivel scăzut.

Mulţi oameni din industrie au realizat faptul că existenţa unui standard pentru reţele fără fir de bandă largă era elementul cheie care lipsea, aşa că organizaţiei IEEE i s-a cerut să alcătuiască un comitet format din oameni din companii cheie şi din mediul academic pentru a schiţa standardul. Următorul număr disponibil în schema de numerotare 802 era **802.16**, deci standardul a primit acest număr. Lucrul a început în iulie 1999, iar standardul final a fost acceptat în aprilie 2002. Oficial, standardul poartă numele “Interfaţă aeriană pentru acces fix la sisteme cu lăţime mare de bandă fără fir”. Totuşi, unii preferă să îl numească **wireless MAN** (**Wireless Metropolitan Area Network**, rom. reţea de întindere metropolitană fără fir) sau **wireless local loop** (rom. buclă locală fără fir). Noi vom privi toţi aceşti termeni ca

fiind interschimbabili.

La fel ca şi alte standarde din seria 802, 802.16 a fost puternic influenţat de modelul OSI, incluzând (sub)nivelurile, terminologia, primitivele de servicii şi altele. Din păcate, ca şi standardul OSI, este destul de complicat. În următoarele secţiuni vom oferi o descriere sumară a lui 802.16, dar tratarea de aici este departe de a fi completă şi lasă neacoperite multe detalii. Pentru informaţii adiţionale despre servicii fără fir de bandă largă în general, vezi (Bolcskei et. al, 2001; şi Webb, 2001). Pentru informaţii specifice despre 802.16, vezi (Eklund et al., 2002).

4.5.1 Comparaţie între 802.11 şi 802.16

În acest moment probabil că vă întrebaţi: De ce să inventăm un nou standard? De ce nu folosim pur şi simplu 802.11? Există câteva motive foarte bune pentru a nu folosi 802.11, în primul rând acela că 802.11 şi 802.16 rezolvă probleme diferite. Înainte de a intra în terminologia lui 802.16, probabil că merită să spunem câteva cuvinte pentru a explica de ce este nevoie de un nou standard.

Mediile în care operează 802.11 şi 802.16 sunt similare în multe privinţe, în primul rând datorită faptului că sunt proiectate pentru a oferi comunicaţii fără fir de mare viteză. Dar de asemenea diferă din multe puncte de vedere majore. Pentru început, 802.16 oferă servicii pentru clădiri, iar clădirile nu sunt mobile. Ele nu migrează dintr-o celulă în alta. Mare parte din 802.11 tratează mobilitatea şi o mare parte dintre aceste aspecte nu sunt relevante aici. Apoi, clădirile dispun de mai multe calculatoare, o complicaţie care nu apare atunci când staţia finală este un singur calculator portabil. Pentru că proprietarii clădirilor sunt de obicei dispuşi să cheltuiască mult mai mulţi bani pe echipamente de comunicaţii decât proprietarii de calculatoare portabile, vor fi disponibile staţii radio mai performante. Această diferenţă înseamnă şi că 802.16 poate folosi comunicaţii cu duplex integral, lucru evitat de 802.11 pentru a menţine preţurile la un nivel scăzut.

Pentru că 802.16 rulează peste o parte dintr-un oraş, distanţele implicate pot fi de mai mulţi kilometri, ceea ce înseamnă că puterea absorbită la staţia de bază poate varia mult de la o staţie la alta. Această variaţie afectează raportul semnal-zgomot, care, la rândul lui, dictează mai multe scheme de modulare. În acelaşi timp, comunicare deschisă peste un oraş înseamnă că securitatea şi confidenţialitatea sunt esenţiale şi obligatorii.

Mai mult, este foarte probabil ca fiecare celulă să aibă mult mai mulţi utilizatori decât o celulă tipică 802.11 şi este de aşteptat ca aceşti utilizatori să consume o lăţime de bandă mult mai mare decât utilizatorii tipici ai 802.11. Până la urmă, nu se întâmplă des ca o companie să invite 50 de angajaţi, cu calculatoare portabile, să se întâlnească într-o singură cameră pentru a vedea dacă se poate satura reţeaua 802.11 prin vizionarea în paralel pe a 50 de filme diferite. Din acest motiv, este nevoie de un spectru mai larg decât oferă banda ISM, forţând 802.16 să opereze într-o gamă de frecvenţe mult mai înalte, de 10-66 GHz, singurul loc în care gama de frecvenţe este încă disponibilă.

Dar aceste unde milimetrice au proprietăţi fizice diferite decât undele mai lungi din ISM, ceea ce duce la necesitatea unui nivel fizic complet diferit. O proprietate a

undelor milimetrice este aceea ca sunt puternic absorbite de apă (în mod special de ploaie, dar până la un anumit nivel şi de către ninsoare, grindină şi, cu puţin ghinion, de ceaţă deasă). În consecinţă, tratarea erorilor este mult mai importantă decât în mediile interioare. Undele milimetrice pot fi focalizate în fascicule direcţionate (802.11 este omnidirecţional), deci soluţiile alese de 802.11 pentru propagarea multicăi sunt contestabile aici.

O altă problemă o constituie calitatea serviciului. Deşi 802.11 oferă suport pentru trafic de timp real (folosind modul PFC), acesta nu a fost proiectat pentru telefonie şi nici pentru utilizare multimedia intensivă. Dimpotrivă, ne aşteptăm ca 802.16 să ofere suport complet pentru aceste aplicaţii, deoarece este gândit să fie folosit atât pentru mediul rezidenţial, cât şi pentru mediul de afaceri.

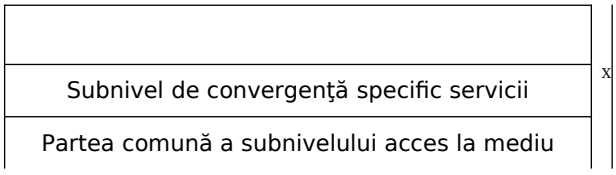
Pe scurt, 802.11 a fost proiectat pentru a fi un Ethernet mobil, pe când 802.16 a fost proiectat pentru a fi o televiziune prin cablu, fără fir, dar staţionară. Aceste diferenţe sunt atât de mari, încât standardele rezultate sunt foarte diferite: ele încearcă să optimizeze lucruri diferite.

O comparaţie foarte sumară cu sistemul de telefonie celulară este de asemenea interesantă. La telefoanele mobile avem de-a face cu staţii mobile de bandă îngustă, orientate pe voce, cu putere scăzută, care comunică folosind unde de lungime medie. Nimeni nu urmăreşte (încă) filme de 2 ore, la rezoluţie înaltă, pe telefoanele GSM. Chiar şi UMTS are speranţe scăzute în a schimba această situaţie. Pe scurt, lumea reţelelor metropolitane fără fir este mult mai solicitantă decât lumea simplă a telefoanelor mobile, deci este necesar un sistem complet diferit. Întrebarea interesantă este dacă 802.16 poate fi folosit pentru dispozitive mobile în viitor. Nu a fost optimizat pentru această utilizare, dar posibilităţi există. Pentru moment, standardul se concentrează pe reţele fixe fără fir.

4.5.2 Stiva de protocoale 802.16

Stiva de protocoale 802.16 este prezentată în fig. 4-31. Structura generală este similară cu cea a celorlalte reţele 802, dar are mai multe subniveluri. Subnivelurile inferioare se ocupă de transmisie. Radioul tradiţional de bandă îngustă foloseşte scheme de modulare convenţionale. Deasupra nivelului de transmisie fizic este un subnivel de convergenţă pentru a ascunde diferitele tehnologii de nivelul legătură de date. De fapt şi 802.11 are ceva asemănător, dar comitetul a decis să nu-l formalizeze cu un nume asemănător OSI.

Cu toate că nu le-am prezentat în figură, două noi protocoale de nivel fizic sunt în lucru deja. Standardul 802.16a va oferi suport pentru OFDM în gama de frecvenţe 2-11 GHz. Standardul 802.16b va lucra în banda ISM de 5 GHz. Ambele constituie încercări de apropiere de 802.11.



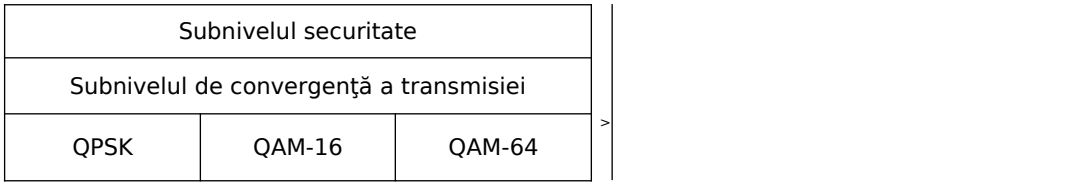


Fig. 4-31. Stiva de protocoale 802.16.

Nivelul legătură de date este constituit din trei subniveluri. Cel mai de jos se ocupă de confidenţialitate şi securitate, aspecte mult mai importante pentru reţelele publice de exterior decât în cadrul reţelelor private de interior. Acest subnivel gestionează cheile, criptarea şi decriptarea.

Următorul este subnivelul comun de acces la mediu. Aici sunt localizate protocoalele principale, cum ar fi cel de gestiune a canalelor. Modelul presupune ca staţia de bază să controleze sistemul. Aceasta poate planifica canalul către abonat (cu baza la abonat) în mod foarte eficient şi joacă de asemenea un rol major în gestionarea canalelor de la abonat (adică abonat către bază). O facilitate neobişnuită a subnivelului de acces la mediu este aceea că, spre deosebire de toate celelalte reţele 802, este complet orientat pe conexiune, pentru a putea garanta calitatea serviciilor pentru comunicaţii multimedia şi pentru telefonie.

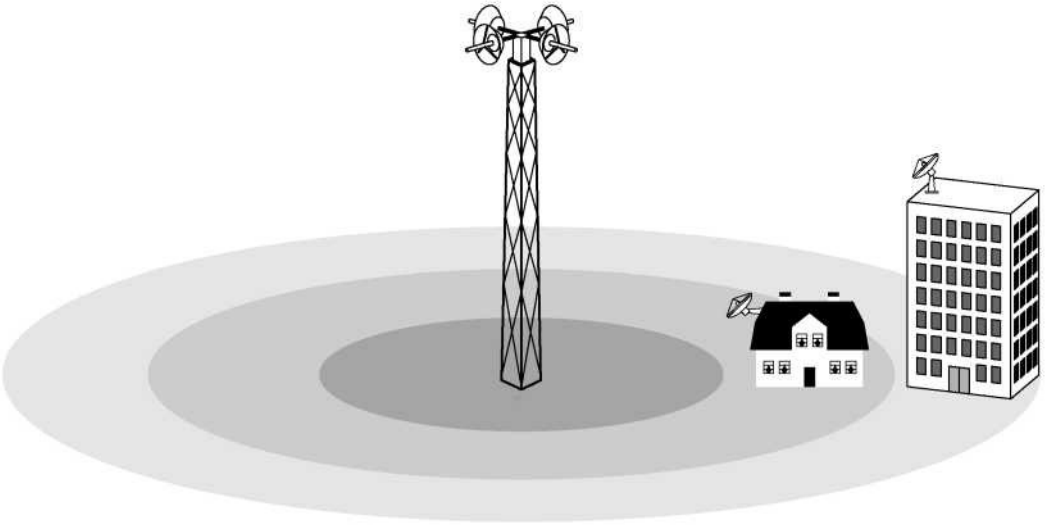
Subnivelul de convergenţă joacă rolul subnivelului de legătură logică din celelalte protocoale 802. Scopul său este interfaţarea cu nivelul reţea. O complicaţie în acest caz este aceea că 802.16 a fost proiectat pentru a integra fără diferenţiere atât protocoale cu datagrame (de ex. PPP, IP şi Ethernet), cât şi ATM. Problema este că protocoalele cu datagrame sunt fără conexiune, pe când ATM-ul este orientat conexiune.

Aceasta înseamnă că fiecare legătură ATM trebuie să fie suprapusă peste o legătură 802.16, în cel mai direct mod posibil. Totuşi, peste care legătură 802.16 ar trebui suprapus un pachet IP recepţionat? Această problemă este rezolvată în cadrul acestui subnivel.

4.5.3 Nivelul fizic 802.16

Aşa cum am menţionat şi mai înainte, transmisia fără fir de bandă largă are nevoie de un spectru larg şi singura posibilitate de a obţine acest spectru este în zona 10-66 GHz. Astfel de unde milimetrice prezintă o proprietate interesantă, care nu există la undele mai lungi: ele se propagă în linii drepte, diferit faţă de sunet, însă foarte

asemănătoare cu lumina. O consecinţă directă este faptul că o staţie de emisie poate avea multiple antene, fiecare îndreptată spre o zonă diferită a ariei de acoperire, aşa cum este prezentat în fig. 4-32. Fiecare sector are proprii utilizatori şi este independent într-o mare măsură de zonele adiacente, fapt care nu este prezent şi în



cazul sistemelor cu celule radio, unde transmisia este în toate direcţiile.

Cum puterea semnalului este milimetrică, acoperirea scade proporţional cu distanţa de la staţia de emisie, raportul semnal/zgomot scade de asemenea, proporţional cu aceeaşi distanţă. Din acest motiv, 802.16 are trei scheme de modulare diferite, în funcţie de cât de departe este staţia abonat de staţia de emisie. Pentru abonaţii foarte apropiaţi de staţia de emisie, este folosit QAM-64, cu 6 biţi/baud. Pentru o distanţă medie, este folosit QAM-16, cu 4 biţi/baud. QPSK este folosit pentru abonaţii aflaţi la o distanţa considerabilă şi suportă 2 biţi/baud. Spre exemplu, pentru un spectru tipic de 25 MHz, QAM-64 oferă 150 Mbps, QAM-16 doar 100 Mbps, în timp ce QPSK atinge până la 50 Mbps. Cu alte cuvinte, cu cât abonatul se află la o distanţă mai mare de staţia de emisie, cu atât viteza de transmisie scade (este asemănător cu ceea ce am văzut la ADSL, fig. 2-27). Diagramele sub formă de constelaţie pentru cele trei tipuri de modulaţie au fost prezentate în fig. 2-25.

Având ca scop crearea unui sistem de bandă largă în prezenţa limitărilor fizice descrise mai sus, designerii lui 802.16 au lucrat din greu la folosirea eficientă a spectrului disponibil. Unul din lucrurile pe care nu le-au apreciat a fost modul în care lucrează sistemele GSM şi DAMPS. Amândouă utilizează, pentru traficul de recepţie şi pentru cel de emisie, benzi de frecvenţă diferite, egale ca dimensiune (simetrice). Dacă pentru voce traficul este simetric în cea mai mare parte, pentru accesul la Internet traficul este mult mai puternic la recepţie decât la transmisie. Din aceste considerente, 802.16 oferă o modalitate mult mai flexibilă de a aloca banda de

QAM-64 (6
QAM-16 (4
QPSK (2

Fig. 4-32. Transmisia în

transfer. Sunt folosite două scheme, **FDD** (**Frequency Division Duplexing**, rom: transmisie duplex prin **d**ivizarea în frecvenţă) şi **TDD** (**Time Division Duplexing – t**ransmisie duplex prin divizarea în timp). A doua schemă este prezentată în fig. 4-33. În acest caz, staţia de emisie trimite periodic cadre. Fiecare cadru conţine unităţi de timp. Primele sunt folosite la traficul de recepţionare. Urmează o perioadă de timp de gardă, folosită de staţii pentru a schimba direcţia de transmisie. În cele din urmă avem intervalul în care se transmit datele locale. Numărul de cuante de timp alocat fiecărui tip de transmisie poate fi modificat dinamic astfel încât banda de transfer să fie folosită cât mai eficient.

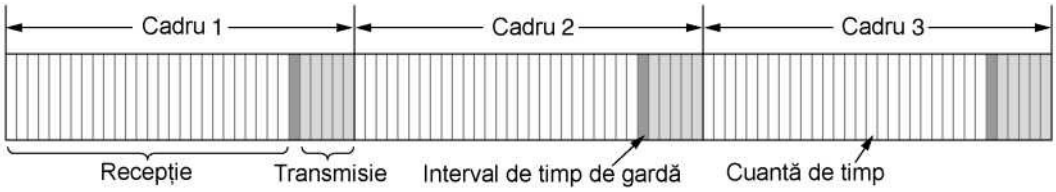


Fig. 4-33. Cadre și intervale de timp în TDD.

Traficul de recepţie este suprapus peste cuantele de timp de către staţia de emisie. Staţia de emisie are controlul deplin pentru acest trafic. Traficul de transmisie este mult mai complex şi depinde de calitatea cerută de serviciul accesat. Vom discuta despre alocarea intervalelor de timp când vom prezenta subnivelul MAC, puţin mai jos.

O altă caracteristică interesantă a nivelului fizic este posibilitatea de a împacheta mai multe cadre MAC într-o singură transmisie fizică. Aceasta creşte eficienţa, prin reducerea numărului total de preambuluri şi antete folosite.

De notat este şi folosirea codurilor Hamming pentru corecţia în avans la nivelul fizic. Aproape toate celelalte tipuri de reţele se bazează pur şi simplu pe trimiterea unor sume de control pentru detectarea erorilor, iar apoi cer retransmisia dacă unul din cadre a fost recepţionat eronat. În cazul transmisiilor de bandă largă sunt estimate atât de multe erori, încât corectarea erorilor se face chiar la nivelul fizic, pe lângă sumele de control de la nivelurile superioare. În ansamblu, efectul pe care îl au toate acestea este de a face canalul să pară mai bun decât este în realitate (în acelaşi mod în care CD-ROM-urile par a fi foarte fiabile, şi toate acestea doar pentru că mai mult de jumătate din numărul total de biţi sunt folosiţi la corectarea de erori la nivelul fizic).

4.5.4 Protocolul subnivelului MAC la 802.16

Nivelul legătură de date este împărţit în trei subniveluri, aşa cum am văzut în fig. 4-31. Cum nu vom studia criptografia înainte de cap. 8, este mai dificil de explicat acum modul de funcţionare a subnivelului de securitate. Este suficient însă să afirmăm că tehnicile de criptare sunt folosite pentru a păstra toate datele transmise secrete. Antetele nu sunt criptate, ci doar informaţia utilă. Aceasta înseamnă că cineva poate să urmărească cine cu cine comunică, dar nu poate înţelege conţinutul mesajelor.

Dacă aveţi deja câteva cunoştinţe despre criptografie, urmează un scurt paragraf

ce explică subnivelul de securitate. Dacă nu cunoaşteţi nimic despre criptografie, este puţin probabil să găsiţi următorul paragraf foarte folositor (însă îl puteţi lua în considerare după ce aţi parcurs cap. 8).

În momentul în care un abonat se conectează la o staţie de emisie, el execută o autentificare mutuală folosind RSA cu cheie publică şi certificatele X.509. Datele utile sunt criptate folosind sistemul de chei simetrice, ori DES triplu, ori folosind înlănţuirea de blocuri cu cifru. Algoritmul AES (Rijndael) e posibil să fie adăugat în curând. Verificările de integritate sunt făcute folosind SHA-1. Nu e aşa că nu a fost chiar atât de greu?

Haideţi să aruncăm o privire peste partea principală a subnivelului MAC. Cadrele MAC ocupă un număr întreg de perioade de timp ale nivelului fizic. Fiecare cadru este format din subcadre, dintre care primele două reprezintă mapările pentru traficul de recepţie şi de transmisie. Aceste mapări relevă ce corespunde fiecărui interval de timp şi ce cuante de timp sunt libere. Maparea pentru recepţie conţine şi un număr de parametri de sistem, pentru a informa noile staţii cu care intră în contact.

Canalul pentru recepţie este destul de simplu, staţia de emisie decide ce conţine fiecare subcadru. Canalul de transmisie este mai complicat deoarece mai mulţi abonaţi intră în competiţie pentru acces. Alocarea acestui canal este legată de problemele de calitate a serviciului. Sunt definite patru clase de servicii:

1. Serviciu cu viteză constantă de transmisie
2. Serviciu pentru aplicaţii de timp real
3. Serviciu pentru aplicaţii care nu necesită timp real
4. Serviciu de tip cea mai bună încercare (best-effort)

Toate serviciile lui 802.16 sunt orientate pe conexiune, iar fiecare dintre conexiuni este stabilită la configurare la unul din tipurile de mai sus. Arhitectura este mult diferită de 802.11 şi Ethernet, care nu prezintă conexiuni la subnivelul MAC.

Serviciul cu viteză constantă de transmisie este folosit pentru transmisie de voce, necompresată, la fel ca pe canalele T1. Acest serviciu are nevoie să trimită o valoare predeterminată de date, la un interval de timp stabilit apriori. Fiecărei conexiuni de acest tip îi este dedicat un număr de intervale de timp. Odată banda de transfer alocată, intervalele de timp sunt puse la dispoziţie automat, fără a mai fi nevoie să fie cerute spre alocare.

Serviciul de aplicaţii de timp real este destinat aplicaţiilor media compresate sau altor aplicaţii software de timp real, în care nevoia de bandă de transfer poate varia. Intră în atribuţiile staţiei de emisie să îl întrebe pe abonat, la un interval fixat de timp, de câtă bandă de transfer are nevoie.

Serviciul pentru aplicaţii ce nu necesită timp real este utilizat pentru transmisii mari de date, cum ar fi transferurile de fişiere mari ca dimensiuni. În acest caz, staţia de emisie interoghează abonatul des, însă nu la intervale fixate de timp. Un abonat care transmite constant date poate cere, prin intermediul cadrelor sale, o interogare din partea staţiei de emisie, pentru a putea trimite date suplimentare.

Dacă o staţie nu răspunde la o interogare repetată de k ori într-un anumit interval de timp, staţia de emisie o va trece pe aceasta într-un grup căruia îi transmite unitar şi îi anulează dreptul la o interogare individuală. Când un astfel de grup este

interogat, oricare dintre staţii poate răspunde, toate intrând în competiţie pentru serviciul cerut. În acest fel, staţiile cu un trafic mic nu consumă inutil interogările staţiei de emisie.

În cele din urmă, serviciul fără garanţie este valabil pentru toate celelalte cazuri. În acest caz nu există interogări, iar abonaţii trebuie să intre în competiţie directă pentru servicii. Cererile de bandă de transfer sunt făcute în acele cuante de timp marcate ca fiind libere în cadrele primite în zona de transmisie pentru conectări. Dacă o cerere a fost satisfăcută, aceasta va fi notată în cadrul următor, în zona de mapare a recepţionărilor. Abonatul va trebui să reîncerce conectarea în caz că cererea nu a fost satisfăcută. Pentru a minimaliza coliziunile este folosit algoritmul de regresie exponenţială de la Ethernet.

Standardul defineşte două forme de alocare a benzii de transfer: pentru staţie şi pentru conexiune. În primul caz, spre exemplu, abonatul face cerere de bandă de transfer în numele tuturor utilizatorilor dintr-o clădire. Când banda de transfer îi este acordată, abonatul va acorda o parte din aceasta fiecărui utilizator, în funcţie de cererile primite de el apriori. În forma a doua de alocare a benzii de transfer (pentru fiecare conexiune), staţia de emisie se ocupă de fiecare conexiune în mod direct.

4.5.5 Structura cadrului 802.16

Toate cadrele MAC încep cu un antet generic. Antetul este urmat, opţional, de datele utile şi tot opţional de o sumă de control (CRC), aşa cum este ilustrat în fig. 4-34. Prezenţa informaţiei utile nu este necesară în cadrele de control, cum sunt de exemplu cadrele pentru alocarea canalelor. Suma de control este şi ea opţională datorită existenţei corecţiei de erori la nivelul fizic şi datorită faptului că nu se încearcă niciodată retransmiterea cadrelor de timp real. Dacă nu se foloseşte retransmisia, de ce să ne complicăm cu o sumă de control?

Biți 1 1 6 1 1 2 1 11 16 8 4
(a) 0 EC Tip CI EK Lungime Identificator de conectare CRC pentru antet Date CRC
Biți 1 1 6 16 16 8
(b) 1 0 Tip Octeți necesari Identificator de conectare CRC pentru antet

Fig. 4-34. (a)Cadru generic. (b)Cadru de cererea a benzii de transmisie.

Urmează o scurtă descriere asupra câmpurilor din fig. 4-34(a). Bitul *EC* arată dacă informaţia utilă este sau nu criptată. Câmpul *Tip* identifică tipul cadrului şi, în esenţă, informează dacă este prezentă împachetarea sau fragmentarea. Câmpul *CI* indică absenţa sau prezenţa sumei de control. În câmpul *EK* se menţionează care dintre cheile de criptare este folosită. Câmpul *Lungime* oferă informaţii despre lungimea completă a cadrului, incluzând şi antetul. Identificatorul de conectare arată cărei conexiuni aparţine acest cadru. În cele din urmă, câmpul CRC pentru antet reprezintă o sumă de control aplicată doar antetului, folosind polinomul $x^8+x2+x+1$.

Al doilea tip de antet, cel din fig. 4-34(b), este folosit pentru cererile de bandă de transfer. El începe cu primul bit de valoare 1, în loc de 0, şi este similar cu cadrul generic, cu excepţia faptului că al doilea şi al treilea octet formează un număr de 16 biţi, care precizează mărimea benzii de transfer necesară. Cadrele de cerere de bandă de transfer nu poartă nici o informaţie utilă şi nici un câmp CRC.

Ar mai fi multe de spus despre 802.16, însă nu mai este loc pentru aceasta. Prin urmare, pentru mai multe informaţii, vă rugăm să consultaţi standardul.

4.6 BLUETOOTH

În 1994, compania L. M. Ericsson a început să fie interesată în a conecta telefoanele mobile pe care le producea cu alte dispozitive (de exemplu, PDA) fără a folosi cabluri. Împreună cu alte patru companii (IBM, Intel, Nokia şi Toshiba) a creat un SIG (Special Interest Group, rom: grup special de interes sau consorţiu) pentru a dezvolta un standard de comunicaţie fără fir pentru interconectarea dispozitivelor de calcul şi comunicare şi a accesoriilor folosind frecvenţe radio pe distanţe scurte care beneficiază de avantajul de a fi o tehnologie ieftină şi fără un consum mare de putere. Proiectul
a fost numit **Bluetooth** după numele regelui viking Harald Blaatand (Bluetooth), ce a ”unificat” (cucerit) Danemarca şi Norvegia, desigur tot fără cabluri.

Deşi ideea iniţială era doar aceea de a scăpa de cablurile dintre dispozitive, încetul cu încetul standardul s-a dezvoltat şi a intrat în scurt timp în aria reţelelor locale fără fir. Pe de o parte, această schimbare are avantajul de a face standardul mai util, dar pe de altă parte se creează o competiţie cu 802.11. Mai mult decât atât, cele două sisteme interferează electric între ele. Este bine de menţionat ca Hewlett-Packard a introdus o reţea infraroşu pentru conectarea fără fire a perifericelor în urmă cu câţiva ani, dar nu a ajuns foarte departe cu aceasta.

Fără a fi descurajat de acest fapt, în iulie 1999, SIG Bluetooth a publicat specificaţia 1.0, având aproximativ 1500 de pagini. La scurt timp, grupul de standardizare IEEE a preluat standardul Bluetooth în aria sa de standarde pentru reţele personale fără fir şi a început să îl ajusteze. Deşi pare ciudat să se încerce să se standardizeze ceva ce a fost deja foarte bine detaliat în specificaţii şi pentru care nu există nici o incompatibilitate care să aibă nevoie de armonizare, istoria arată că redactarea unui standard de către o entitate neutră, aşa cum este IEEE, a dus de cele mai multe ori la promovarea tehnologiei respective. Pentru a fi mai exacţi, trebuie menţionat că specificaţia Bluetooth este elaborată pentru întregul sistem, de la nivelul fizic şi până la nivelul aplicaţie. Comitetul 802.15 standardizează numai nivelul fizic şi nivelul de legătură de date, restul stivei de protocoale nefiind în afara atenţiei sale.

Chiar dacă IEEE a aprobat în 2002 primul standard pentru reţele personale 802.15.1, SIG Bluetooth este încă foarte activ la îmbunătăţirea standardului. În ciuda faptului că versiunile emise de SIG Bluetooth şi IEEE nu sunt identice, se speră că în curând cele două vor converge spre un standard comun.

4.6.1 Arhitectura Bluetooth

Haideţi să începem studiul nostru asupra sistemului Bluetooth cu o scurtă prezentare de ansamblu asupra a ceea ce conţine şi ce este proiectat să facă. Unitatea de bază a unui sistem Bluetooth este un **piconet** (**pico reţea**) format dintr-un nod stăpân (master) şi până la 7 noduri sclav (slave), toate într-o regiune cu diametrul maxim de 10 metri. Mai multe piconet-uri pot exista în aceeaşi încăpere şi chiar pot fi conectate printr-un nod de trecere, aşa cum arată fig. 4-35. O colecţie interconectată de piconet-uri este denumită **scatternet** (**reţea dispersată**).

În afara celor şapte noduri sclav active dintr-un piconet, în reţea pot exista până la 255 de noduri în modul parcat. Acestea sunt dispozitive pe care stăpânul le-a trecut în modul de consum redus, economisind astfel consumul de putere de la baterii. În modul parcat, un dispozitiv nu poate face altceva decât să răspundă la semnalele temporare ale stăpânului sau la cele de activare. De asemenea, există încă două stări intermediare: aşteptare şi „ascultare” (eng. sniff), însă ele nu vor fi discutate în această parte.

Motivul pentru care s-a ales arhitectura stapân/sclav este acela că designerii au intenţionat să faciliteze o implementare completă a cipurilor Bluetooth sub 5$. Consecinţa acestei decizii este aceea că sclavii sunt destul de limitaţi, şi, în esenţă, ei fac doar ceea ce le spune stăpânul să facă. Ca principiu de bază, un piconet este un sistem TDM centralizat, în care stăpânul controlează ceasul şi determină care dispozitiv primeşte dreptul de a comunica şi pentru ce perioadă de timp. Toate comunicaţiile sunt între stăpân şi sclav; nu pot exista comunicaţii directe sclav-sclav.

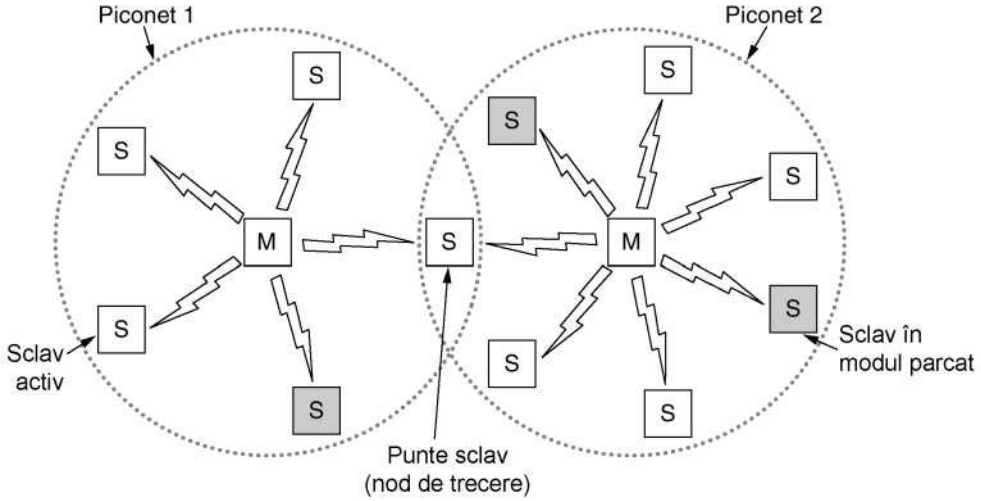


Fig. 4-35. Doua piconet-uri pot fi conectate să formeze o reţea dispersată (scatternet).

4.6.2 Aplicaţii Bluetooth

Majoritatea protocoalelor de reţea oferă doar canale între entităţile comunicante şi lasă designe- rii de aplicaţii să decidă în ce mod vor să le folosească. Spre exemplu, 802.11 nu specifică dacă utilizatorii ar trebui să folosească notebook-urile personale pentru a-şi citi emailul, pentru a naviga pe internet, sau orice altceva. Spre deosebire de acestea, specificaţia Bluetooth V1.1 defineşte 13 aplicaţii specifice care sunt suportate şi oferă stive diferite de protocoale pentru fiecare dintre ele. Din păcate, această abordare este destul de complexă şi de aceea în această carte vom încerca să simplificăm anumite aspecte. Cele 13 aplicaţii, denumite **profiluri**, sunt enumerate în fig. 4-36. Uitându-ne sumar la ele, putem să determinăm mult mai uşor care sunt intenţiile SIG Bluetooth.

Denumire	Descriere
Acces generic	Proceduri pentru întreţinerea legăturii
Descoperire de servicii	Protocol de descoperire a serviciilor oferite
Port serial	Înlocuitor pentru cablul de port serial
Intershimbare generică a	Defineşte relaţia client-server pentru vehicularea de
Acces la reţeaua locală	Protocol între un calculator mobil şi o reţea fixă
Reţea pe linie telefonică	Oferă posibilitatea ca un notebook să se apeleze folosind un telefon mobil
Fax	Permite unui fax mobil să comunice cu un telefon
Telefonie fără fir	Conectează un set de căşti de staţia sa locală de
Emitator-Receptor portabil	Radio-telefon portabil digital
Căşti de telefon cu transmiţător	Permite transmisii de voce hands-free (fără folosirea
Trimitere a obiectelor	Oferă o modalitate de schimbare a obiectelor simple
Transfer de fişiere	Oferă o facilitate mai generală de transfer de fişiere
Sincronizare	Oferă posibilitatea unui PDA să se sincronizeze cu un

Fig. 4-36. Profiluri Bluetooth.

Profilul de acces generic nu este o aplicaţie în sine, ci este mai degrabă o bază pe

care se pot construi aplicaţiile reale. Principala sa atribuţie este să ofere posibilitatea de a crea şi a menţine conexiuni (canale) sigure între stăpân şi sclav. De asemenea, profilul de descoperire a serviciilor este relativ generic. El este folosit de dispozitive pentru a descoperi ce servicii sunt oferite de către alte staţii din reţea. Toate dispozitivele Bluetooth trebuie să implementeze aceste două profiluri. Celelalte rămân opţionale.

Profilul de port serial este un protocol de transport care este folosit de majoritatea celorlalte profiluri. Acesta emulează o linie serială şi este foarte util în special pentru aplicaţiile mai vechi care necesit o astfel de facilitate.

Profilul de interschimbare generică a obiectelor defineşte o relaţie client-server pentru transmisia de date. Clienţii iniţiază operaţii, însă un sclav poate fi sau client sau server. Ca şi profilul de port serial, este un punct de pornire pentru celelalte profiluri.

Următorul grup de trei profiluri este pentru lucrul în reţea. Profilul de acces la reţeaua locală permite unui dispozitiv Bluetooth să se conecteze la o reţea specificată. Acest profil este un competitor direct cu 802.11. Profilul de dial-up de reţea a fost motivaţia iniţială a întregului proiect. El permite ca un notebook să se conecteze fără fire la un telefon mobil ce conţine un modem intern. Profilul de fax este similar cu cel de dial-up de reţea, cu diferenţa că permite ca un fax neconectat la reţeaua de telefonie să trimită şi să primească faxuri folosind telefoane mobile, fără existenţa unui fir între cele două.

Următoarele trei profiluri sunt pentru telefonie. Profilul de telefonie fără fir oferă o modalitate de a conecta un set de căşti fără fir la staţia de telefon. Momentan, majoritatea telefoanelor fără fir nu pot fi folosite şi ca telefoane mobile, dar în viitor, este posibil ca telefoanele mobile şi cele fără fir să devină unul şi acelaşi lucru. Profilul Emiţător-Receptor portabil (intercom) permite ca două telefoane să fie interconectate ca radiotelefoane portabile (walkie-talkie). În sfârşit, profilul pentru căşti oferă comunicaţii de voce între căşti cu transmiţător şi staţia de bază a telefonului, spre exemplu, pentru comunicaţii fără intervenţie manuală (hands-free) în timpul condusului maşinii.

Ultimele trei profiluri sunt folosite pentru schimburi de obiecte între două dispozitive fără fir. Acestea ar putea fi cărţi de vizită, poze sau fişiere cu date. În particular, profilul de sincronizare este adresat încărcării de date pe un PDA sau notebook când părăseşte locuinţa şi colectarea de date de la acesta la revenire.

Era oare cu adevărat necesar să fie menţionate toate aceste aplicaţii în detaliu şi să fie create diferite stive de protocoale pentru fiecare dintre ele? Poate că nu era necesar, însă au existat diferite grupuri de lucru care au divizat standardul în părţi mai mici şi fiecare dintre ele şi-a concentrat eforturile spre rezolvarea unei probleme specifice şi a creat propriul profil. Gândiţi-vă la aceasta ca la o aplicaţie pentru legea lui Conway (în aprilie 1968, în revista Datamation, Melvin Conway observa că atunci când propui la n persoane să scrie un compilator, primeşti ca soluţie un compilator cu n- treceri de compilare, sau şi mai general, structura software oglindeşte structura grupului care l-a produs). Era deci posibil să fie finalizate doar două stive de protocoale în loc de treisprezece, unul pentru transferul de fişiere şi unul pentru fluidizarea comunicaţiilor real-time.

4.6.3 Stiva de protocoale Bluetooth

Standardul Bluetooth conţine mai multe protocoale grupate în niveluri. Structura nivelurilor nu respectă modelul OSI, modelul TCP/IP, modelul 802 sau oricare alt model existent. Totuşi, IEEE lucrează la modificarea acestuia astfel încât să urmeze cât mai bine tiparul 802. Arhitectura de bază a protocolului Bluetooth, aşa cum a fost modificată de comitetul 802, este prezentată în fig. 4-37.

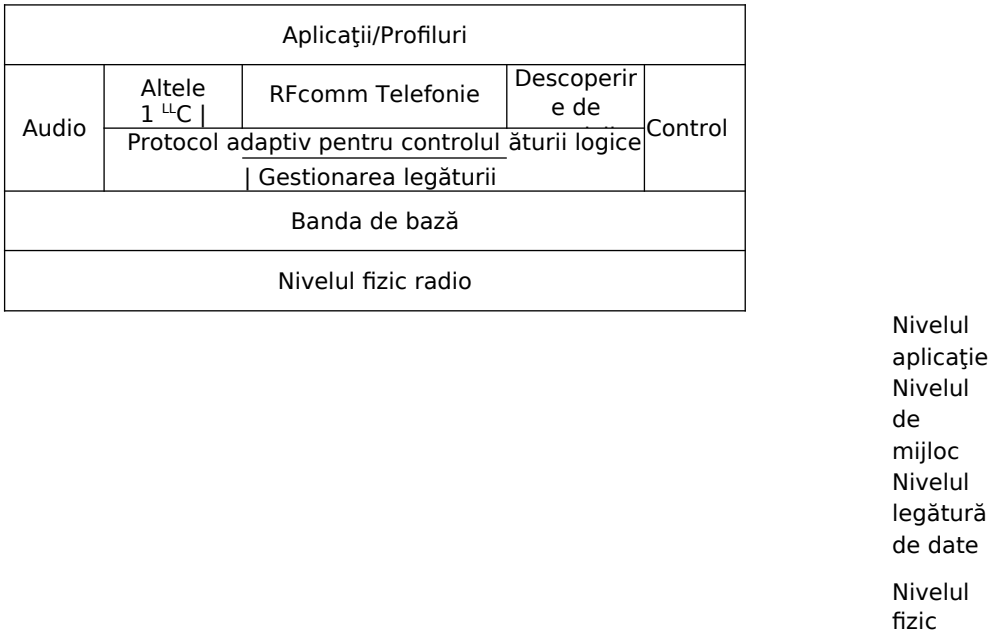


Fig. 4-37. Versiunea 802.15 a arhitecturii protocolului Bluetooth.

Nivelul de bază este reprezentat de nivelul fizic radio, care corespunde destul de bine nivelului fizic din modelele OSI şi 802. El se ocupă cu transmisia radio şi modularea semnalului. Atenţia, în cadrul acestui nivel, s-a focalizat pe modelarea unui sistem ieftin astfel încât acesta să devină rapid un produs de larg consum. Nivelul bandă de bază este întrucâtva analog cu subnivelul MAC, însă include elemente de nivel fizic. El se ocupă de modul în care stăpânul controlează unităţile de timp şi cum acestea sunt grupate în cadre.

Apoi urmează un nivel cuprinzând un grup de protocoale relativ asemănătoare. Gestionarul legăturii se ocupă cu stabilirea de canale logice între dispozitive, incluzând consumul de putere, autentificarea şi calitatea serviciilor. Protocol adaptiv pentru controlul legăturii logice (adesea numit L2CAP – Logical Link Control Adaptation Protocol, rom: protocol de adaptare a controlului legăturii logice) oferă o interfaţă nivelurilor superioare prin ascunderea detaliilor de transmisie. El este analog cu subnivelul LLC din standardul 802, însă, din punct de vedere tehnic, este diferit de acesta. Aşa cum sugerează şi numele, protocolul de control şi cel audio se ocupă de control şi respectiv, de partea audio. Aplicaţiile pot apela direct la acesta fără a avea nevoie de intermedierea protocolului L2CAP.

Nivelul următor este nivelul de mijloc şi acesta conţine o îmbinare de protocoale diferite. Nivelul LLC 802 a fost inserat aici de către IEEE pentru compatibilitatea sa cu celelalte reţelele 802. Protocoalele RFcomm, de telefonie şi serviciul de descoperire sunt native. RFcomm (Radio Frequency communication, rom: comunicare pe frecvenţe radio) este protocolul care emulează portul serial standard pe care îl găsim la orice PC obişnuit, pentru interconectarea de tastatură, mouse şi modem, precum şi pentru alte dispozitive. El a fost dezvoltat pentru a permite dispozitivelor perimate să îl folosească în continuare cu uşurinţă. Protocolul de telefonie este un protocol de timp real folosit pentru profilurile orientate pe 3 legături. El se ocupă şi de stabilirea şi de terminarea conexiunii. În cele din urmă, protocolul de descoperire a serviciilor este folosit pentru găsirea serviciilor din reţea.

Nivelul cel mai înalt este cel în care sunt localizate aplicaţiile şi profilurile. Ele se folosesc de protocoalele din nivelurile inferioare pentru a îşi realiza sarcinile. Fiecare aplicaţie are un subset propriu dintre aceste protocoale. Dispozitivele specifice, cum sunt căştile, conţin exclusiv acele protocoale necesare aplicaţiei. În secţiunile următoare vom studia cele mai de jos trei niveluri din stiva de protocoale Bluetooth, deoarece acestea corespund în oarecare măsură cu subnivelurile fizic şi MAC.

4.6.4 Nivelul Bluetooth radio

Nivelul radio transmite biţii de la stăpân la sclav sau vice-versa. Este un sistem de putere mică cu o rază de acoperire de 10 metri, în banda de 2.4 GHz ISM. Banda este divizată în 79 de canale de 1 MHz fiecare. Modularea se face prin deplasarea în frecvenţă a cheilor, cu un bit pe Hz, realizând în total o rată de transfer de date de 1 Mbps, însă o mare parte din aceasta este consumată de supraîncărcare. Pentru a aloca rezonabil canalele, se foloseşte metoda FHSS (salturi de frecvenţă într-un spectru larg) cu 1600 salturi/sec şi cu intervalul de timp de 62 ps. Toate nodurile dintr-un piconet rulează simultan, stăpânul dictând secvenţa de rulare.

Deoarece şi 802.11 şi Bluetooth operează în banda ISM de 2.4 GHz, pe aceleaşi 79 de canale, ele interferează unul cu celalalt. Întrucât rata de eşantionare a timpului Bluetooth este mult mai mare decât la 802.11, este mult mai probabil ca un dispozitiv Bluetooth să întrerupă transmisia unui 802.11 decât invers. Dat fiind faptul că 802.11 şi 802.15 sunt amândouă standarde IEEE, organizaţia caută o soluţie la această problemă, dar nu este deloc simplu ştiind că ambele sisteme folosesc banda ISM din acelaşi motiv: aceasta nu are nevoie de o licenţă de utilizare. Standardul 802.11a foloseşte cealaltă lungime de bandă ISM (5 GHz), însă are o rază mult mai scurtă decât cea a lui 802.11b (datorită fenomenelor fizice legate de undele radio), aşadar folosirea 802.11a nu este o soluţie foarte bună în toate cazurile. Câteva companii au rezolvat problema dând vina pe Bluetooth. O soluţie de marketing ar fi ca reţeaua cu mai multă putere (politică şi economică, nu electrică) să ceară părţii mai slabe să îşi modifice standardul, pentru a nu mai interfera cu ea. Câteva idei în legătură cu această problemă sunt oferite în (Lansford et. co., 2001).

4.6.5 Nivelul bandă de bază Bluetooth

Nivelul bandă de bază este cea mai apropiată legătură pe care Bluetooth o are cu

subnivelul MAC. Acesta transformă o secvenţa de biţi într-un cadru şi defineşte câteva formate de bază. În forma cea mai simplă, stăpânul din fiecare piconet defineşte o serie de cuante de timp de 625 ps; transmisia stăpânului se desfăşoară în cuantele de timp pare, iar transmisia sclavilor în intervalele de timp impare. Aceasta este metoda tradiţională de diviziune multiplexată a timpului, cu stăpânul preluând jumătate din perioadele de timp şi sclavii împărţind cealaltă jumătate. Cadrele pot ocupa 1, 3 sau 5 cuante de timp.

În schema de alimentare sunt preconizate 250-260 ps pentru fiecare salt, pentru a putea permite stabilizarea circuitelor radio. Setări mai rapide decât atât sunt posibile numai la un cost mai ridicat. Pentru un singur cadru, după calibrare, 366 din cei 625 de biţi sunt pierduţi. Din aceştia, 126 sunt folosiţi la codul de acces şi la antet, lăsând ceilalţi 240 de biţi pentru date. Când cadrele ocupă cinci cuante de timp, este necesară o singură perioadă de stabilizare a circuitelor, aşa că din 5 x 625 = 3125 de biţi în cinci intervale de timp, 2781 sunt disponibili pentru nivelul bandă de bază. În concluzie, cadrele mai lungi sunt mult mai eficiente decât cele scurte, formate pe o singură cuantă de timp.

Fiecare cadru este transmis pe un canal logic, numit **legătură** (**link**), între stăpân şi sclav. Există două tipuri de legături. Primul tip este denumit legătură **ACL** (**Asynchronous Connection-Less,** rom: Asincron fără conexiune) şi este utilizat pentru comutarea de pachete de date ce sunt disponibile la intervale neregulate de timp. Aceste date sunt primite de la nivelul L2CAP la transmiţător şi sunt emise către nivelul L2CAP de la receptor. Traficul ACL este realizat în metoda „cea mai bună încercare” (best-effort). Nu sunt oferite nici un fel de garanţii. Cadrele pot fi pierdute şi atunci este necesară retransmiterea lor. Un sclav poate avea cel mult o legătură ACL la stăpânul său.

Celălalt tip este denumit legătură **SCO (Synchronous Connection Oriented,** rom: sincron orientat pe conexiune**)**, fiind folosit pentru date reale, cum sunt transmisiile telefonice. Acestui tip de canal îi este alocat un număr fix de cuante de transmisie în fiecare direcţie. Datorită naturii critice a legăturilor SCO, cadrele trimise pe aceste legături nu sunt niciodată retransmise. În schimb, pot fi

folosite mecanisme de corecţie în avans a erorilor pentru a avea un grad mai mare de încredere. Un sclav poate avea până la trei legături SCO către stăpânul său. Fiecare legătură SCO poate transmite 64.000 bps PCM pe canal audio.

4.6.6 Nivelul L2CAP Bluetooth

Nivelul L2CAP are trei funcţii majore. Prima funcţie este aceea de a accepta pachete până la 64 KB de la nivelurile superioare şi de a le sparge în cadre pentru transmisie. La sfârşit, cadrele sunt reasamblate în pachete.

A doua funcţie este de a multiplexa şi demultiplexa pachete provenite de la diverse surse. Când un pachet a fost reasamblat, nivelul L2CAP determină cărui protocol superior îi este adresat, spre exemplu RFcomm sau telefonic.

A treia funcţie este aceea de a garanta calitatea serviciilor cerute, atât în timpul realizării conexiunii cât şi în timpul operaţiilor obişnuite. De asemenea, la configurare este negociat şi maximul de informaţie utilă permisă, pentru a preveni situaţia în care un dispozitiv ce generează pachete mari suprasolicită un dispozitiv care foloseşte pachete mai mici. Această funcţie este necesară deoarece nu toate dispozitivele pot utiliza pachete de 64 KB.

4.6.7 Structura cadrului Bluetooth

Există mai multe tipuri de cadre, cel mai important este reprezentat în fig. 4-38. El începe cu un cod de acces care identifică de obicei stăpânul; astfel, dacă un sclav se află în raza radio a doi stăpâni, va putea să ştie cărui stăpân se adresează. Următorul câmp este un antet de 54 de biţi, incluzând câmpurile tipice ale subnivelului MAC. Apoi urmează câmpul de date, de maxim 2744 de biţi (pentru o transmisie de 5 intervale de timp). Pentru o singură unitate de timp formatul este acelaşi, cu excepţia

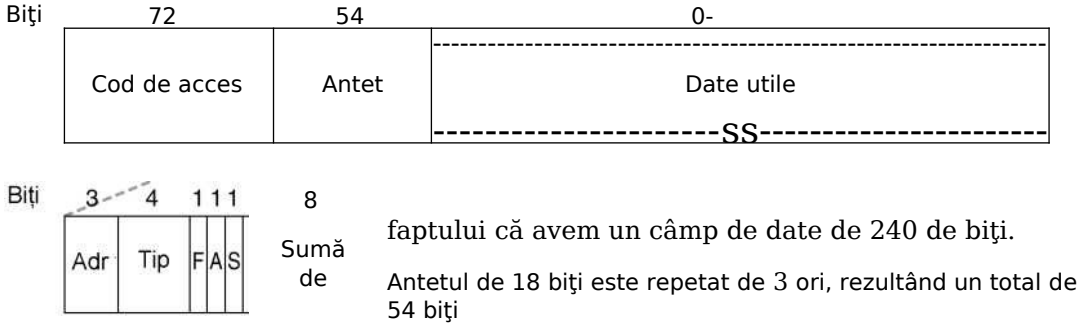


Fig. 4-38. Cadru de date Bluetooth tipic.

Să aruncăm acum o scurtă privire peste antet. Dintre cele opt dispozitive active, câmpul adresă îl identifică pe acela pentru care este destinat cadrul. Câmpul tip identifică tipul cadrului (ACL, SCO, interogare sau vid), tipul de corecţie a erorii ce va fi folosit în câmpul de date şi lungimea cadrului. Bitul de revărsare (flow) este folosit

de către un sclav când memoria sa tampon este plină şi, prin urmare, nu mai poate primi date. Bitul de confirmare pozitivă este folosit pentru a valida un cadru recepţionat corect. Bitul de secvenţă este folosit pentru a numerota cadrele şi este utilizat la retransmisii. Protocolul este de tipul pas-cu-pas, aşa că un singur bit este suficient. Apoi urmează un câmp
de sumă de control de 8 biţi. Tot antetul de 18 biţi este repetat de trei ori pentru a forma antetul de 54 de biţi din fig. 4-38. La receptor, un circuit simplu examinează cele trei copii ale fiecărui bit. Dacă toate sunt identice, bitul este acceptat. Dacă nu sunt identice, majoritatea va decide. Aşadar 54 de biţi din totalul de biţi transmisibili sunt folosiţi pentru a propaga antetul de 10 biţi. Pentru că se doreşte obţinerea unei transmisii de date de încredere într-un mediu plin de interferenţe, utilizând un dispozitiv ieftin, cu consum scăzut (2.5 mW) şi cu capacitate redusă de calcul, se impune un nivel mare de redundanţă în date.

Pentru câmpul de date din cadrele ACL sunt folosite mai multe forme. Totuşi cadrele SCO sunt foarte simple: câmpul de date este mereu de 240 de biţi. Trei variante sunt definite permiţând o informaţie utilă de 80, 160 şi 240 de biţi, restul fiind utilizat pentru corecţia de erori. În varianta cea mai de încredere (80 biţi de informaţie utilă), conţinutul este repetat de trei ori, la fel ca şi antetul.

Deoarece un sclav nu poate folosi decât cuantele de timp impare, el primeşte 800 intervale de timp/sec, la fel ca şi stăpânul. Cu o informaţie utilă de 80 de biţi, capacitatea canalului de la sclav este 64000 bps, iar capacitatea canalului de la stăpân este tot de 64000 bps, suficient pentru un singur canal de voce PCM duplex-integral (acesta este motivul pentru care a fost aleasă o rată a salturilor de 1600 salturi/sec). Aceste date arată că un canal duplex-integral de voce cu 64.000 bps în ambele direcţii, folosind cel mai de încredere format saturează complet un piconet, în ciuda unei benzi de transfer de 1 Mbps. Pentru varianta cu o încredere foarte mică (240 biţi/interval de timp, fără redundanţă la acest nivel), trei canale duplex-integral sunt suportate simultan, acesta fiind motivul pentru care un număr maxim de trei legături SCO sunt permise la un sclav.

Sunt multe lucruri de adăugat despre Bluetooth, însă din păcate nu avem un spaţiu suficient aici. Pentru informaţii suplimentare, vă recomandăm (Bhagwat, 2001; Bisdikian, 2001; Bray şi Sturman, 2002; Haartsen, 2000; Johansson şi colab, 2001; Miller şi Bisdikian, 2001; Sairam şi colab., 2002).

4.7. COMUTAREA LA NIVELUL LEGĂTURII DE DATE

Multe organizaţii au mai multe LAN-uri şi doresc să le conecteze. LAN-urile pot fi conectate prin dispozitive numite **punţi** (**bridges**), care operează la nivelul legăturii de date. Punţile examinează adresele de la nivelul legăturii de date pentru a face rutarea. Întrucât ele nu trebuie sa examineze câmpurile cu informaţie utilă ale cadrelor pe care le rutează, ele pot transporta pachete IPv4 (utilizate acum în

Internet), IPv6 (vor fi utilizate în Internet în viitor), AppleTalk, ATM, OSI, sau orice alt fel de pachete. Spre deosebire de punţi, ruterele examinează adresele din pachete şi fac rutarea pe baza acestora. Deşi aceasta pare o departajare clară între punţi şi rutere, câteva îmbunătăţiri de ultimă ora, precum apariţia Ethernetului comutat, au complicat şi mai mult lucrurile, cum vom vedea mai târziu. În capitolele care urmează ne vom ocupa de punţi şi comutatoare, în special pentru conectarea diferitelor LAN-uri 802. Pentru o tratare cuprinzătoare a punţilor, comutatoarelor şi a altor subiecte înrudite, vezi (Perlman,1992).

Înainte de a intra în tehnologia punţilor, merită să aruncăm o privire asupra câtorva situaţii obişnuite în care acestea sunt folosite. Vom menţiona şase motive pentru care o singură organizaţie poate ajunge să aibă LAN-uri multiple. În primul rând, multe universităţi şi departamente ale unor corporaţii au propriile lor LAN-uri, în principal pentru a-şi conecta calculatoarele personale, staţiile de lucru şi serverele. Deoarece scopurile departamentelor diferă, departamente diferite vor alege

LAN-uri diferite, separat de ceea fac alte departamente. Mai devreme sau mai târziu, este nevoie de interacţiune, deci este nevoie de punţi. În acest exemplu, LAN-urile multiple au apărut datorită autonomiei proprietarilor lor.

În al doilea rând, organizaţia poate fi răspândită geografic în mai multe clădiri separate aflate la distanţe considerabile. Poate fi mai ieftină soluţia cu LAN-uri separate în fiecare clădire, conectate prin punţi şi legături în infraroşu, decât soluţia cu întinderea unui singur cablu coaxial pe întreaga suprafaţă.

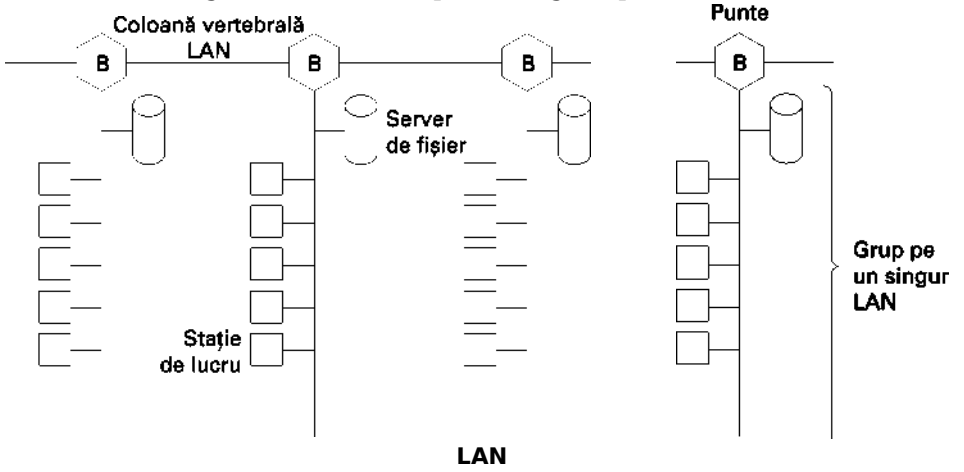


Fig. 4-39. LAN-uri multiple conectate printr-o coloană vertebrală pentru a trata un trafic total mai mare decât capacitatea unui singur LAN.

În al treilea rând, pentru a face faţă traficului, poate fi necesară spargerea unei entitaţi care din punct de vedere logic constituie un singur LAN în LAN-uri separate. La multe universităţi, de exemplu, mii de staţii de lucru sunt disponibile pentru profesori şi studenţi. Fişierele sunt ţinute de obicei pe servere şi sunt încărcate pe maşinile utilizatorilor la cerere. Dimensiunea mare a acestui sistem împiedică punerea tuturor staţiilor de lucru pe un singur LAN – lărgimea de bandă totală necesară este mult prea mare. În schimb sunt folosite LAN-uri multiple conectate prin punţi, după cum este arătat în fig. 4-39. Fiecare LAN conţine un grup de staţii de lucru cu propriul său server de fişiere, astfel încât cea mai mare parte a traficului este limitată la un singur LAN şi astfel nu se încarcă suplimentar coloana vertebrală.

Merită amintit faptul că deşi figurăm LAN-urile ca având acces la un acelaşi mediu de comunicaţie ca în fig. 4-39 (abordarea clasică), ele sunt cel mai frecvent implementate cu noduri sau, în zilele noastre, mai ales cu comutatoare. Oricum, un mediu de transmisie comun cu numeroase maşini conectate la el şi un nod care conectează maşini sunt identice din punct de vedere funcţional. În ambele cazuri, toate maşinile aparţin aceluiaşi domeniu de coliziuni şi toate utilizează protocolul CSMA/CD pentru a trimite cadre. Cum am văzut şi mai înainte şi cum vom vedea din nou în curând, LAN-urile comutate sunt diferite.

În al patrulea rând, există anumite situaţii în care un singur LAN ar fi potrivit în ceea ce priveşte traficul, dar distanţa fizică între cele mai îndepărtate calculatoare

este prea mare (de exemplu, mai mult de 2.5 km pentru Ethernet). Chiar dacă este uşor de întins cablul, reţeaua nu ar funcţiona din cauza întârzierilor excesiv de mari pentru propagarea dus/întors a semnalelor. Singura soluţie este partiţionarea LAN-ului şi instalarea de punţi între segmente. Folosind punţile, poate fi mărită distanţa fizică totală acoperită.

În al cincilea rând, trebuie considerată problema siguranţei. Pe un singur LAN, un nod defect, care trimite tot timpul un şir continuu de date alterate, va compromite LAN-ul. Punţile pot fi inserate în puncte critice pentru a preveni ca un singur nod care funcţionează defectuos să afecteze întregul sistem. Spre deosebire de un repetor, care doar copiază ceea ce vede, o punte poate fi programată să exercite un anumit control privind ceea ce trimite mai departe şi ceea ce nu trimite.

În al şaselea (şi ultimul) rând, punţile pot contribui la securitatea organizaţiei. Cele mai multe interfeţe LAN au un mod **transparent de lucru** (promiscuous mode), în care *toate* cadrele sunt transferate calculatorului, nu numai cele care sunt adresate acestuia. Spionilor şi băgăreţilor le place acest lucru. Prin inserarea punţilor în diferite locuri şi prin grija de a nu transmite traficul de date sensibile, este posibilă izolarea unor părţi din reţea, astfel încât datele să nu ajungă în mâinile cui nu trebuie.

Ideal ar fi ca punţile să fie perfect transparente, aceasta însemnând că ar fi posibilă mutarea unei maşini de pe un segment de cablu pe un altul fără a schimba nimic în hardware, în software, sau în tabelele de configurare. De asemenea, ar trebui să fie posibil ca o maşina de pe oricare segment să comunice cu maşini în oricare alt segment fără a ţine seamă de tipul LAN-urilor folosite în cele două segmente sau în segmentele dintre ele. Acest ţel este uneori atins, dar nu totdeauna.

4.7.1 Punţi de la 802.x la 802.y

După ce am văzut de ce sunt necesare punţile, să ne întoarcem la felul în care funcţionează acestea. Fig. 4-40 ilustrează funcţionarea unei punţi simple, dublu-port. Gazda A intr-un LAN fără fir (802.11) are un pachet de trimis către gazda fixă B într-un Ethernet (802.3) la care LAN-ul fără fir este conectat. Acest pachet coboară la subnivelul LLC şi dobândeşte un antet LLC (figurat cu negru). Apoi trece la subnivelul MAC şi îi este ataşat un antet 802.11 (ca de altfel şi o încheiere, care nu este figurată). Această structură este transmisă apoi prin aer şi ajunge în cele din urmă la staţia de la bază; aceasta observă că trebuie să trimită structura într-un LAN Ethernet. Apoi ajunge la puntea care conectează reţeaua 802.11 de reţeaua 802.3 la nivelul fizic şi îşi continuă drumul spre nivelurile superioare.

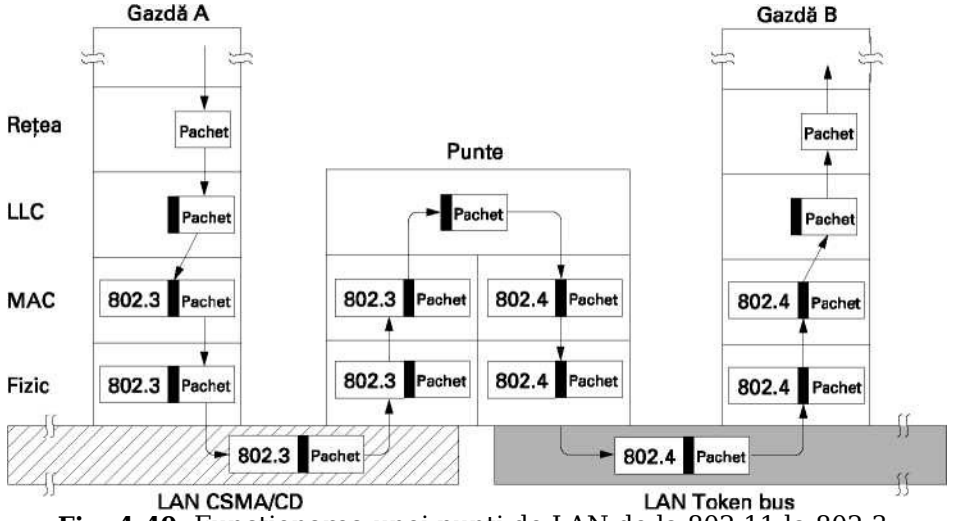


Fig. 4-40. Funcţionarea unei punţi de LAN de la 802.11 la 802.3.

În subnivelul MAC din punte, antetul 802.11 este îndepărtat. Pachetul simplu (cu antetul LLC) este predat subnivelului LLC din punte. În acest exemplu, pachetul este destinat unei subreţele
802.3 conectată la punte, astfel încât îşi face drum pe partea 802.3 a punţii şi pleacă mai departe în Ethernet. De notat că o punte conectând k LAN-uri diferite va avea k subniveluri MAC diferite şi k niveluri fizice diferite, câte unul pentru fiecare tip.

Până acum, din câte am văzut, transmiterea unui pachet de la un LAN la altul pare simplă, însa nu este cazul. În această secţiune vom scoate în evidenţă câteva din dificultăţile întâlnite atunci când se încearcă construirea unei punţi între LAN-uri 802 diferite (şi MAN-uri). Ne vom concentra atenţia asupra 802.3, 802.11 şi 802.16, dar mai sunt şi altele cu seturile lor unice de probleme.

Pentru început, fiecare dintre LAN-uri foloseşte un format de cadru diferit (vezi fig. 4-41). Faţă de diferenţele dintre Ethernet, token bus şi token ring, care apăreau datorita egourilor marilor corporaţii şi din cauza motivelor istorice, în acest caz, diferenţele sunt bine argumentate. De exemplu, câmpul Durată la 802.11 este acolo datorită protocolului MACAW şi nu are nici un sens în Ethernet. Prin urmare, orice transfer între LAN-uri diferite cere reformatare, ceea ce consumă timp de procesor, necesită o nouă calculare a sumei de control şi introduce posibilitatea erorilor nedetectate datorată biţilor eronaţi în memoria punţii.

O a doua problemă este că LAN-urile interconectate nu funcţionează neapărat la aceeaşi rată de transfer. Atunci când se transmite un şir lung de cadre concatenate de la un LAN rapid la unul mai lent, puntea nu va putea transmite cadrele în ritmul în care sosesc. De exemplu, dacă un gigabit Ethernet varsă biţi într-un 11-Mb 802.11b LAN la viteză maximă, puntea va trebui să memoreze traficul, în speranţa ca va avea memorie suficienta. Punţile care conectează trei sau mai multe LANuri au o problemă similară în cazul în care mai multe LAN-uri încearcă să alimenteze acelaşi LAN de ieşire în acelaşi moment chiar dacă toate LAN-urile au aceeaşi viteză.

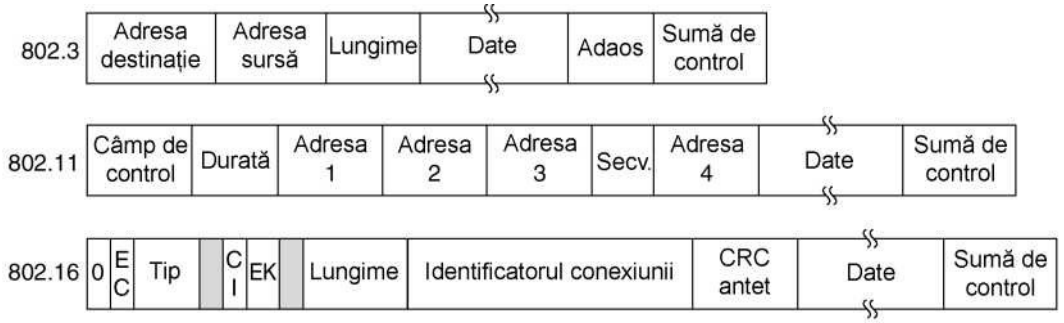


Fig. 4-41. Formatele cadrelor IEEE 802. Desenul nu este la scară.

O a treia, şi potenţial cea mai serioasă problemă dintre toate, este că diferite LAN-uri 802 au o lungime maximă de cadru diferită. O problemă evidentă apare atunci când un cadru lung trebuie transmis unui LAN care nu îl poate accepta. La acest nivel, împărţirea cadrului iese din discuţie. Toate protocoalele presupun recepţionarea totală sau deloc a cadrelor. Nu există posibilitatea de reasamblare a cadrelor din unităţi mai mici. Aceasta nu înseamnă că asemenea protocoale nu ar putea fi inventate. Ele pot fi şi au fost. Doar că nici un protocol legătură de date nu are această caracteristică, aşa că punţile nu trebuie să se atingă de informaţia utilă din cadru. Fundamental, nu există nici o soluţie. Cadrele care sunt prea lungi pentru a fi transmise trebuie eliminate. Cam atât în ceea ce priveşte transparenţa.

Un alt punct este securitatea. 802.11 şi 802.16 suportă criptarea la nivelul legăturii de date. Ethernet nu suportă. Aceasta înseamnă că diversele servicii de criptare disponibile la reţelele fără fir sunt pierdute când traficul trece printr-o reţea Ethernet. Încă şi mai rău, dacă o staţie fără fir utilizează criptare la nivelul legăturii de date, nu există nici o modalitate de a decripta datele când ajung în Ethernet. Dacă o staţie fără fir nu utilizează criptarea, traficul acesteia este expus de-a lungul legăturii prin aer. În ambele situaţii există o problemă.

O soluţie la problema securităţii ar fi criptarea la un nivel superior, dar în acest fel o staţie 802.11 trebuie să afle dacă vorbeşte cu o altă staţie într-o reţea 802.11 (însemnând că foloseşte criptare la nivelul legăturii de date) sau nu (însemnând că nu foloseşte). Forţarea unei staţii să facă o alegere distruge transparenţa.

Punctul final este calitatea serviciilor. Atât 802.11 cât şi 802.16 o oferă în forme diverse, primul folosind modul PCF şi ultimul folosind conexiuni cu rată de transfer constantă. Ethernet-ul nu oferă nimic în acest sens, aşa că traficul de la oricare dintre ceilalţi va pierde din calitate atunci când trece printr-o reţea Ethernet.

4.7.2 Interconectarea locală a reţelelor

Capitolul anterior s-a ocupat de problemele întâmpinate la conectarea printr-o singură punte a două LAN-uri IEEE 802 diferite. Oricum, în organizaţiile mari cu multe LAN-uri, numai simpla interconectare a acestora ridică mai multe probleme, chiar şi dacă toate LAN-urile sunt Ethernet. Ideal, ar trebui să fie posibil să te duci şi să cumperi punţi proiectate după standardul IEEE, să le conectezi şi totul să funcţioneze perfect, instantaneu. Nu ar trebui să fie nevoie de modificări de har-

dware, de modificări de software, de setarea adreselor, de încărcarea tabelelor sau parametrilor, de nimic altceva. Se conectează numai cablurile şi funcţionează. Mai mult, funcţionarea LAN-urilor existente nu ar trebui să fie afectată în nici un fel de punţi. Cu alte cuvinte punţile ar trebui să fie complet transparente (invizibile pentru hardware şi software). Destul de surprinzător, chiar au reuşit. Haideţi să vedem cum se realizează această magie.

O punte transparentă operează în mod transparent (promiscuous mode), acceptând orice cadru transmis pe oricare dintre LAN-urile la care este ataşată. De exemplu, să considerăm configuraţia din fig. 4-42. Puntea B1 este conectată la LAN-urile 1 şi 2, iar puntea B2 este conectată la LAN-urile 2, 3 şi 4. Un cadru destinat lui A de la LAN 1 care ajunge la puntea B1 poate fi eliminat imediat, pentru că este deja pe LAN-ul care trebuie, dar un cadru care ajunge de la LAN 1 pentru C sau F trebuie transmis.

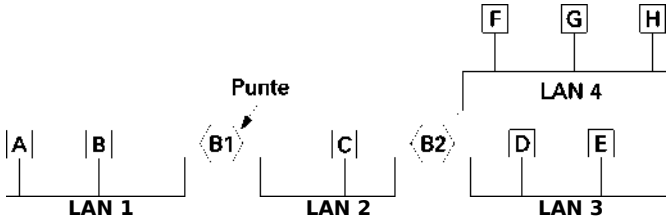


Fig. 4-42. O configuraţie cu patru LAN-uri şi două punţi.

La sosirea unui cadru, o punte trebuie să decidă dacă să îl elimine sau să îl transmită mai departe, iar dacă îl transmite, către ce LAN să îl trimită. Această decizie este luată căutând adresa destinaţiei într-o tabelă de dispersie menţinută în interiorul punţii. Tabelul poate să includă fiecare destinaţie posibilă şi cărei linii de ieşire (de fapt, cărui LAN) îi aparţine. De exemplu, tabelul lui B2 ar include A ca aparţinând lui LAN 2, din moment ce tot ce trebuie să ştie B2 este către care LAN să trimită cadrele destinate lui A. Nu prezintă interes faptul că ulterior vor avea loc mai multe transmisii.

La prima conectare a punţilor, toate tabelele de dispersie sunt vide. Nici una dintre punţi nu ştie unde se află destinaţiile, astfel încât toate folosesc algoritmul de inundare: orice cadru care vine pentru o destinaţie necunoscută este trimis către toate LAN-urile la care este conectată puntea, cu excepţia celui din care a venit. Cu trecerea timpului, punţile află unde se găsesc destinaţiile, după cum este descris în cele ce urmează. Odată ce o destinaţie este cunoscută, cadrele destinate ei sunt puse pe LAN-ul care trebuie, în loc să fie inundate.

Algoritmul folosit de punţile transparente se numeşte *învăţare regresivă (backward learning).* După cum a fost menţionat anterior, punţile lucrează în mod transparent (promiscuous), astfel încât toate văd fiecare cadru trimis pe oricare dintre LAN-urile lor. Uitându-se la adresa sursei, ele pot afla care calculator este accesibil pe care LAN. De exemplu, dacă puntea B1 din fig. 4-38 vede un cadru din LAN 2 venind de la C, ştie că staţia C trebuie să fie accesibilă prin LAN 2 şi creează o intrare în tabela de dispersie, în care notează că pentru cadrele care merg la C ar trebui să folosească LAN 2. Orice cadru ulterior adresat lui C care vine din LAN 1 va fi transmis mai

departe, pe când un cadru pentru C venit din LAN 2 va fi abandonat.

Topologia se poate schimba după cum calculatoarele şi punţile sunt în funcţiune sau nu, sau mutate de colo-colo. Pentru a trata topologii dinamice, de câte ori se creează o intrare în tabela de dispersie, în ea este notat timpul de sosire a cadrului. De câte ori soseşte un cadru a cărui destinaţie se află deja în tabel, intrarea sa este adusă la zi cu timpul curent. Astfel, timpul asociat fiecărei intrări arată ultimul moment în care a fost primit un cadru de la respectivul calculator.

Periodic, un proces din punte scanează tabela de dispersie şi curăţă toate intrările mai vechi de câteva minute. În acest fel, dacă un calculator este scos din LAN-ul său, plimbat prin clădire şi reinstalat în altă parte, în câteva minute va reveni la funcţionarea normală, fără vreo intervenţie manuală. Acest algoritm semnifică de asemenea că dacă un calculator este inactiv pentru câteva minute, orice trafic trimis spre el va trebui inundat, până când calculatorul respectiv va trimite un cadru.

Procedura de dirijare pentru un cadru sosit depinde de LAN-ul din care soseşte (LAN-ul sursă) şi de LAN-ul în care se află destinaţia sa (LAN-ul destinaţie), după cum urmează:

1. Dacă LAN-ul sursă este acelaşi cu LAN-ul destinaţie, abandonează cadrul.
2. Dacă LAN-ul sursă şi cel destinaţie sunt diferite, transmite cadrul.
3. Dacă LAN-ul destinaţie nu este cunoscut, foloseşte inundarea.

Acest algoritm trebuie aplicat pentru fiecare cadru care soseşte. Există cipuri VLSI speciale care realizează căutarea şi actualizarea în tabela de dispersie, doar în câteva microsecunde.

4.7.3 Punţi cu arbore de acoperire

Pentru a mări siguranţa, unele locaţii folosesc două sau mai multe punţi în paralel între perechi de LAN-uri, aşa cum este arătat în fig. 4-43. Totuşi, acest aranjament introduce şi unele probleme suplimentare, întrucât creează bucle în topologie.

Un simplu exemplu al acestor probleme poate fi văzut în fig. 4-43, observând modul în care este tratat cadrul F cu destinaţie necunoscută. Fiecare punte, urmând regulile obişnuite pentru tratarea destinaţiilor necunoscute, foloseşte inundarea care, în acest exemplu, nu înseamnă decât copierea cadrului pe LAN 2. Puţin după aceea, puntea 1 vede F2, un cadru cu destinaţie necunoscută, pe care îl copiază pe LAN 1, generând F3 (care nu este arătat în figură). La fel, puntea 2 copiază F1 pe LAN1 generând F4 (care nu este arătat). Acum puntea 1 trimite F4 şi puntea 2 copiază F3. Acest ciclu se continuă la nesfârşit.

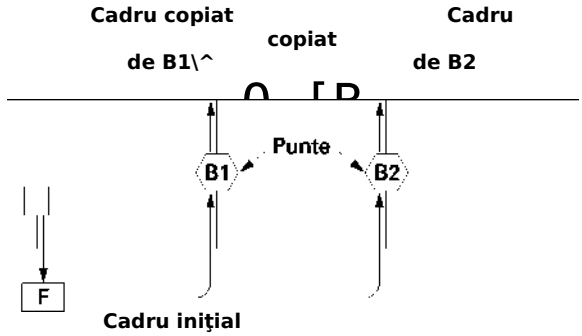


Fig. 4-43. Două punţi transparente paralele

Soluţia acestei probleme este ca punţile să comunice unele cu altele şi să suprapună peste topologia actuală un arbore de acoperire care ajunge la fiecare LAN. De fapt, în interesul construirii unei topologii fictive fără bucle, sunt ignorate câteva conexiuni posibile între LAN-uri. De exemplu, în fig. 4- 44(a) apar nouă LAN-uri interconectate prin zece punţi. Această configuraţie poate fi rezumată într- un graf cu LAN-urile drept noduri. Un arc leagă oricare două LAN-uri care sunt conectate de o punte.

Graful poate fi redus la un arbore de acoperire renunţând la arcurile figurate ca linii punctate în fig. 4-44(b). Folosind acest arbore de acoperire, există un singur drum de la fiecare LAN la fiecare alt LAN. Odată ce punţile s-au înţeles asupra arborelui de acoperire, toată transmiterea dintre LAN-uri urmăreşte arborele de acoperire. Din moment ce există un drum unic de la fiecare sursă la fiecare destinaţie, buclele sunt imposibile.

Pentru a construi arborele de acoperire, punţile trebuie să aleagă mai întâi o punte care va reprezenta rădăcina arborelui. Ele fac această alegere prin emiterea de către fiecare punte a numărului de serie, instalat de fabricant, garantat ca fiind unic în întreaga lume. Puntea cu cel mai mic număr serial devine rădăcină. Apoi se construieşte un arbore de drumuri minime de la rădăcină la fiecare punte şi LAN. Acest arbore este un arbore de acoperire. Dacă o punte sau un LAN cade, trebuie calculat un nou arbore de acoperire.

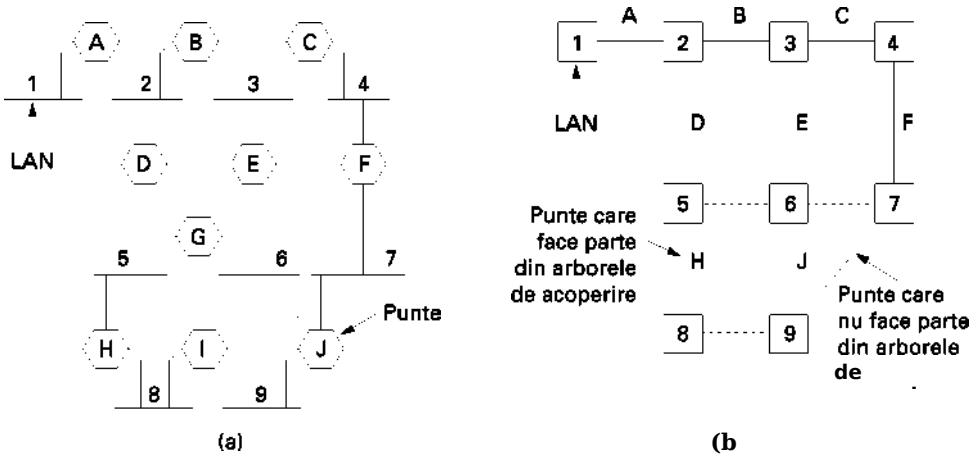


Fig. 4-44. (a) LAN-uri interconectate. (b) Arbore de acoperire pentru LAN-uri. Liniile punctate nu fac parte

Rezultatul acestui algoritm este că se stabileşte un drum unic de la fiecare LAN la rădăcină, şi astfel la fiecare alt LAN. Deşi arborele acoperă toate LAN-urile, nu neapărat toate punţile sunt prezente în arbore (pentru a evita buclele). După ce a fost stabilit arborele de acoperire, algoritmul continuă să ruleze pentru a detecta automat schimbări în topologie şi a actualiza arborele. Algoritmul distribuit, folosit pentru construirea arborelui de acoperire, a fost inventat de Perlman şi este descris în detaliu în (Perlman, 1992). Acesta este standardizat în IEEE 802.1D.

4.7.4 Punţi aflate la distanţă

Punţile sunt de cele mai multe ori folosite pentru conectarea a două (sau mai multe) LAN-uri aflate la distanţă unele de altele. De exemplu, o companie poate avea fabrici în mai multe oraşe, fiecare dintre acestea cu propriul său LAN. Ideal ar fi ca toate aceste LAN-uri să fie interconectate pentru ca sistemul în întregime să funcţioneze ca un mare LAN.

Acest ţel poate fi atins punând câte o punte fiecărui LAN şi conectând punţile în perechi cu linii punct-la-punct (de exemplu linii închiriate de la o companie de telefoane). Un sistem simplu, cu trei LAN-uri, este prezentat în fig. 4-45. Aici se aplică algoritmul de dirijare obişnuit. Cel mai simplu este să se privească cele trei linii punct-la-punct ca LAN-uri fără gazde. Adică, un sistem obişnuit de şase LAN-uri interconectate prin patru punţi. Nimic din ce am studiat până acum nu spune că un LAN trebuie să aibă gazde.

Pe liniile punct-la-punct pot fi folosite diverse protocoale. O posibilitate este alegerea unui protocol de legătură de date punct-la-punct standard, cum ar fi PPP, punând cadre MAC complete în câmpul de informaţie utilă. Această strategie funcţionează cel mai bine dacă LAN-urile sunt identice şi singura problemă este transmiterea cadrelor la LAN-ul care trebuie. Altă posibilitate este eliminarea antetului şi a încheierii cadrelor MAC la puntea sursă, punând ceea ce a mai rămas în câmpul de informaţie utilă al protocolului punct-la-punct. Un nou antet şi o nouă încheiere MAC pot fi apoi generate la puntea destinaţie. Un dezavantaj al acestei abordări este că suma de control care ajunge la puntea destinaţie nu este cea calculată de gazda sursă, existând posibilitatea ca erori cauzate de biţi eronaţi în memoria unei punţi să nu fie detectaţi.

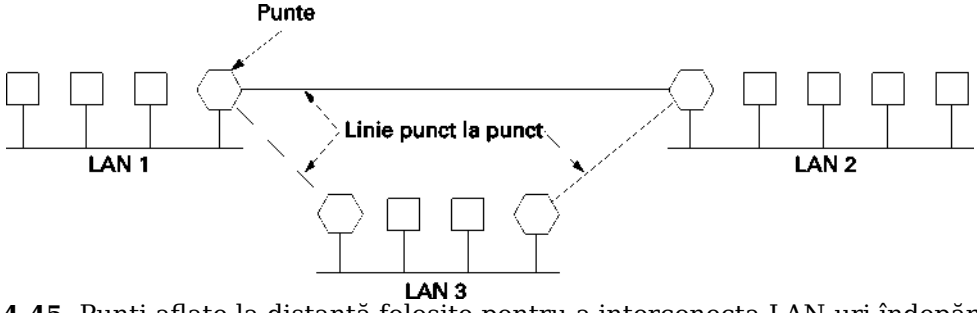


Fig. 4-45. Punţi aflate la distanţă folosite pentru a interconecta LAN-uri îndepărtate.

4.7.5 Repetoare, Noduri, Punţi, Comutatoare, Rutere şi Porţi

Până acum în aceasta carte am văzut a mulţime de feluri de a transfera cadre şi pachete de pe un segment de cablu pe altul. Am amintit de repetoare, noduri, punţi, comutatoare, rutere şi porţi. Toa

te aceste dispozitive sunt utilizate în mod curent, dar ele diferă mai mult sau mai puţin unul de altul. Deoarece sunt atât de multe, merită sa le analizăm împreună pentru a vedea asemănările şi diferenţele dintre ele.

Pentru început, aceste dispozitive operează la niveluri diferite, cum este ilustrat în fig. 4-46(a). Nivelul contează pentru că diferitele dispozitivele folosesc segmente diverse din informaţie pentru a decide cum să comute. Într-un scenariu tipic, utilizatorul creează date pentru a fi trimise către o maşina aflată la distanţă . Aceste date sunt trimise nivelului transport, unde li se adaugă un antet, de exemplu un antet TCP, şi se transmite rezultatul mai jos către nivelul reţea. Nivelul reţea adaugă propriul antet pentru a forma un pachet pentru nivelul reţea, de exemplu un pachet IP. În fig. 4- 46(b) observăm pachetul IP colorat în gri. Apoi pachetul ajunge la nivelul legăturii de date, care îi adaugă propriul antet şi suma de control (CRC) şi trimite cadrul rezultat către nivelul fizic pentru transmisie, de exemplu intr-un LAN.

Acum să ne uităm la dispozitivele de comutare şi să vedem legătura lor cu pachetele şi cadrele. La cel mai de jos nivel, nivelul fizic, se află repetoarele. Acestea sunt dispozitive analogice ce sunt conectate intre două segmente de cablu. Un semnal ce apare pe unul din aceste cabluri este amplificat şi trimis pe celălalt cablu. Repetoarele nu înţeleg cadrele, pachetele sau antetele. Ele înţeleg doar tensiuni electrice. Ethernetul clasic, de exemplu, a fost proiectat sa permită folosirea a patru repetoare în scopul de a extinde lungimea maximă a cablului de la 500 de metri la 2500 de metri.

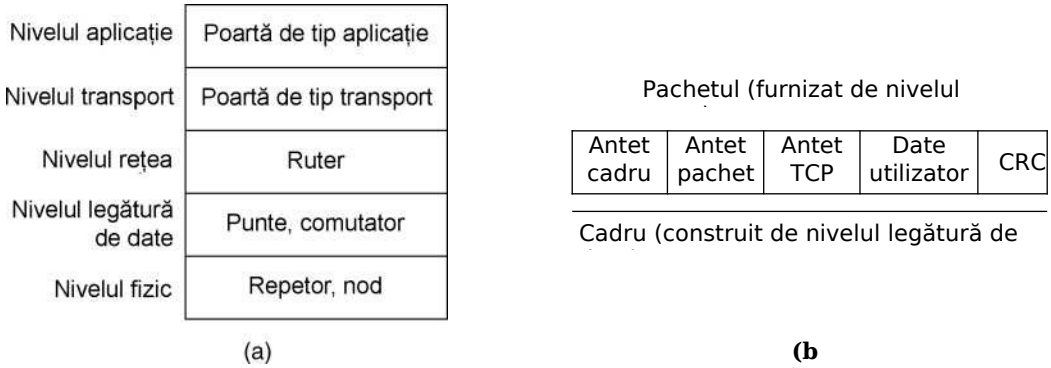


Fig. 4-46. (a) Corespondenţa dintre niveluri şi dispozitive. (b) Cadre,

În continuare ajungem la noduri. Un nod are un număr de linii de intrare pe care le uneşte din punct de vedere electric. Cadrele care ajung la nod pe oricare linie sunt trimise afară pe toate celelalte liniile. Dacă două cadre ajung în acelaşi timp se vor ciocni la fel ca şi atunci când ar fi transmise pe un cablul coaxial. Cu alte cuvinte un nod formează un singur domeniu de coliziune. Toate liniile ce intră în nod trebuie să lucreze la aceeaşi viteză. Nodurile diferă de repetoare prin faptul ca (de obicei) nu amplifică semnalele pe care le primesc şi sunt proiectate pentru a suporta multe plăci de extensie cu mai multe intrări; totuşi, diferenţele nu sunt semnificative. Ca şi repetoarele, nodurile nu examinează adresele 802 şi nici nu le utilizează în vreun fel. Un nod este arătat în fig. 4-47(a).

Acum vom aborda nivelul legăturii de date, unde găsim punţile şi comutatoarele.

Tocmai am studiat punţile. O punte conectează două sau mai multe LAN-uri aşa cum este arătat în fig. 4-47(b). Când un cadru ajunge, software-ul din punte extrage adresa destinaţie din cadru şi caută în tabela sa vadă unde să trimită cadrul. Pentru Ethernet, această adresă este adresa destinaţie de 48 de biţi prezentată în fig. 4-17. Asemănător unui nod, o punte moderna are placi de extensie, de obicei pentru patru sau opt intrări de un anumit tip. O placă de extensie pentru Ethernet nu poate manevra, sa zicem, cadre token ring, pentru că nu ştie unde să găsească adresa destinaţie în antetul cadrului. Oricum, o punte poate avea placi de extensie pentru diferite tipuri de reţele şi diferite viteze. Spre deosebire de nod, la punte fiecare linie se află în propriul domeniu de coliziune.

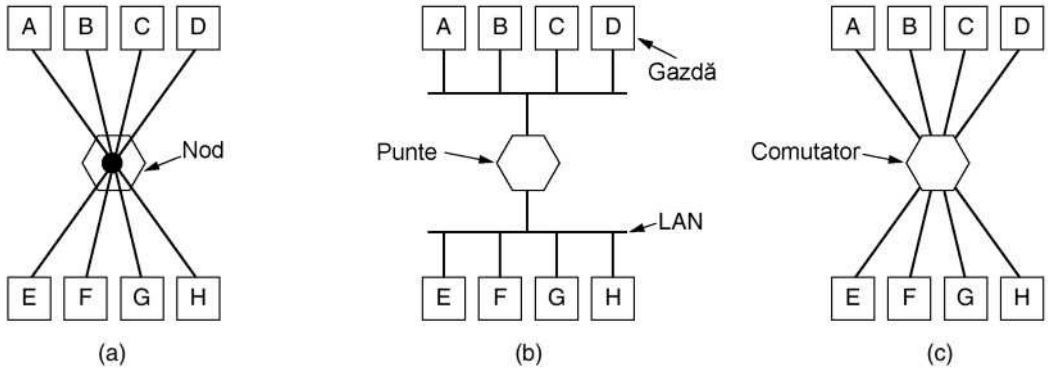


Fig. 4-47. (a) Un nod. (b) O punte. (c) Un comutator.

Comutatoarele sunt similare cu punţile deoarece amândouă rutează cadre pe baza adreselor. De fapt, mulţi oameni folosesc aceste două denumiri fără a face o distincţie clară intre termeni. Principala diferenţă este aceea că un comutator este cel mai adesea folosit pentru a conecta calculatoare individuale, aşa cum este arătat în fig. 4-47(c). Ca o consecinţă, când gazda A în fig. 4-47(b) doreşte sa trimită un cadru către gazda B, puntea primeşte cadrul dar nu îl ia în considerare. Din contră, după cum se vede în fig. 4-47(c), comutatorul trebuie sa retransmită cadrul de la A la B deoarece nu există alt drum pentru ca acest cadru să ajungă. Întrucât fiecare port al comutatorului este de obicei conectat la un singur calculator, comutatorul trebuie să aibă loc pentru mai multe placi de extensie decât punţile care trebuie să conecteze numai reţele. Fiecare placă de extensie are un spaţiu tampon pentru cadrele recepţionate. Deoarece fiecare port se află în propriul domeniu de coliziune, comutatoarele nu pierd niciodată cadre din cauza coliziunilor. Totuşi, dacă un comutator primeşte cadre mai repede decât le poate retransmite, este posibil ca în scurt timp sa nu mai aibă memorie tampon liberă şi sa înceapă să arunce din cadrele primite.

Pentru a relaxa puţin problema, comutatoarele moderne încep sa retransmită cadre imediat ce antetul destinaţie ajunge, dar înainte ca restul cadrului sa ajungă (bineînţeles asigurându-se ca linia de ieşire este disponibila). Aceste comutatoare nu utilizează tehnica de memorare şi retransmitere. Câteodată ele sunt menţionate drept comutatoare cu transmitere de fragmente (cut-through switches). De obicei acest tip de comutator este implementat în întregime în hardware, în timp ce tradiţionalele punţi conţin un CPU ce face comutare cu memorare şi retransmitere la nivel software.

Dar deoarece toate punţile şi comutatoarele moderne conţin circuite integrate speciale pentru comutare, diferenţele dintre comutatoare şi punţi ţin mai mult de probleme de marketing decât de probleme tehnice.

Până acum, am văzut repetoare şi noduri, care sunt foarte asemănătoare, precum şi punţi şi comutatoare, care sunt de asemenea foarte asemănătoare între ele. Acum trecem mai departe la rutere, care sunt şi ele diferite de cele menţionate mai sus. Când un pachet ajunge la un ruter, antetul şi sfârşitul cadrului sunt eliminate şi pachetul localizat în informaţia utila a cadrului (înnegrit în fig. 4-46) trece către software-ul de rutare. Acest software foloseşte antetul pachetului pentru a alege o linie de ieşire. Pentru un pachet IP, antetul pachetului va conţine adrese de 32 de biţi (IPv4) sau adrese de 128 de biţi (IPv6), în nici un caz adrese 802 de 48 de biţi. Software-ul de rutare nu vede adresele cadrelor şi nici măcar nu ştie daca pachetul a venit de pe un LAN sau de pe o linie punct la punct. În cap. 5 vom studia ruterele şi rutarea.

Mai sus cu un nivel găsim porţile de transport (gateways). Acestea conectează doua calculatoare ce utilizează diferite protocoale de transport orientate pe conexiune. De exemplu, să presupunem că un calculator care utilizează protocolul TCP/IP orientat pe conexiune, trebuie să discute cu un calculator care foloseşte protocolul ATM orientat pe conexiune. Poarta de transport poate copia pachete de la o conexiune la alta, refăcând pachetele după necesităţi.

În încheiere, porţile la nivelul aplicaţie înţeleg formatul şi conţinutul datelor şi traduc mesajul de la un format la altul. De exemplu, o poartă de poşta electronică poate traduce mesaje Internet în mesaje SMS pentru telefoane mobile.

4.7.6 LAN-uri virtuale

La începutul dezvoltării reţelelor locale de calculatoare, cabluri groase galbene şerpuiau prin conductele de cablu ale multor clădiri de birouri. Acestea conectau toate calculatoarele pe la care treceau. Adesea erau mai multe cabluri conectate la coloana vertebrală centrală (ca în fig. 4-39) sau la un nod central. Nu se acorda nici o importanţă corespondenţei între calculatoare şi LAN-uri. Toţi oamenii din birouri alăturate erau conectaţi la acelaşi LAN idiferent dacă aparţineau sau nu aceleiaşi organizaţii. Poziţionarea fizică a calculatoarelor domina logica.

Totul s-a schimbat odată cu apariţia lui 10Base-T şi a nodurilor în anii 1990. Clădirile au fost recablate (cu costuri considerabile) pentru a scoate vechile cabluri galbene, instalându-se în loc cabluri cu perechi de fire torsadate de la fiecare birou până la locurile de conectare centrală de la capătul fiecărui coridor sau până la camera unde se afla calculatorul principal, aşa cum este ilustrat în fig. 4-48.

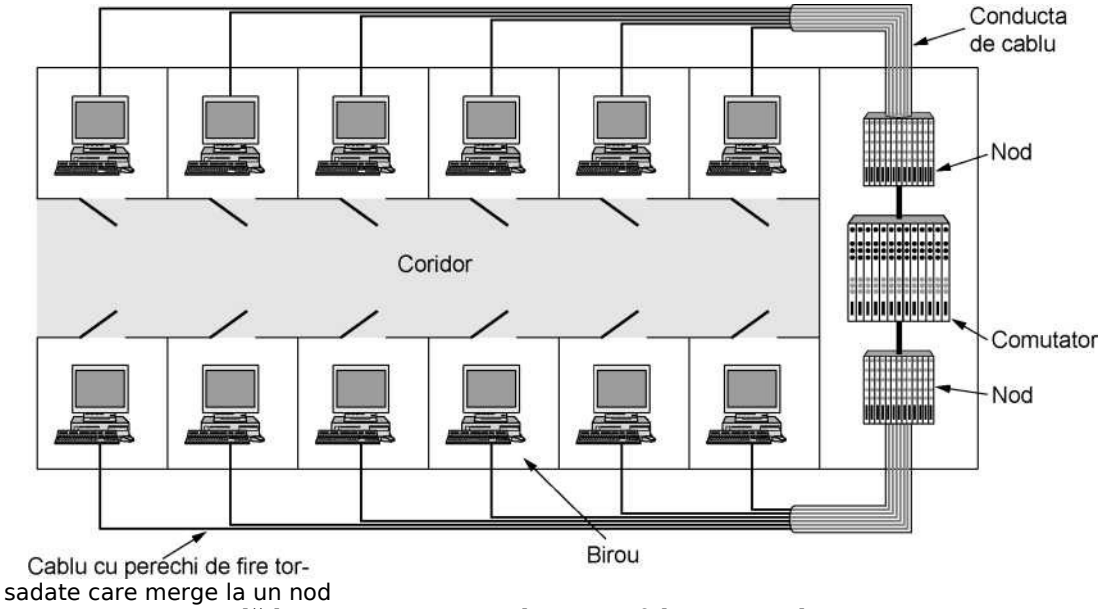


Fig. 4-48. O clădire cu reţea centralizata ce foloseşte noduri şi un comutator.

Dacă vicepreşedintele care se ocupa de cablare era vizionar, se instalau cabluri cu perechi de fire torsadate de categoria a cincea; dacă acesta era un statistician era utilizat cablu de telefon existent (categoria 3) (pentru a fi înlocuit câţiva ani mai târziu când a apărut Ethernet-ul rapid).

Folosindu-se noduri (şi mai târziu, comutatoare) Ethernet, era adesea posibilă configurarea LANurilor din punct de vedere logic mai mult decât din punct de vedere fizic. Daca o companie doreşte k LAN-uri, cumpără k noduri. Alegând cu grijă ce conectoare vor fi introduse în nod, ocupanţii LANului pot fi aleşi din punct de vedere organizaţional fără a se da prea mare importanţă amplasării geografice. Bineînţeles, dacă doi oameni aparţinând aceluiaşi departament lucrează în clădiri diferite, cel mai probabil aceştia sunt conectaţi în noduri diferite şi astfel în LAN-uri diferite. Cu toate acestea, situaţia se prezintă mult mai bine decât în cazul LAN-urilor bazate pe amplasare geografică.

Contează cine este conectat şi în ce LAN? Oricum, în toate organizaţiile, toate LAN-urile sunt în cele din urmă interconectate. Pe scurt, răspunzând la întrebare: adesea contează. Dintr-o multitudine de motive administratorii de reţea îşi doresc să grupeze utilizatorii în LAN-uri pentru a reflecta structura organizatorică mai degrabă decât structura fizică a clădirii. O problemă este securitatea. Orice interfaţă de reţea poate fi configurată în mod transparent, copiind tot traficul care soseşte pe canalul de comunicaţie. Multe departamente, cum ar fi cele de cercetare, patente şi contabilitate, deţin informaţii pe care nu doresc să le facă cunoscute în afara departamentului. În situaţii ca acestea, punerea tuturor oamenilor din departament într-o singură reţea locală fără a permite vreunui fel de trafic să iasă din această reţea este o soluţie bună. Totuşi, administratorul va vrea sa audă despre astfel de aranjamente doar dacă toţi oamenii din fiecare departament sunt localizaţi în birouri adiacente, fără birouri interpuse între acestea.

Poate să apară şi o a doua problemă. Unele reţele locale sunt utilizate mai intensiv

decât altele şi poate fi benefică separarea acestora la anumite momente. De exemplu, dacă persoanele de la cercetare rulează tot felul de experimente dichisite care din când în când scapă de sub control şi le saturează reţeaua locală, persoanele de la contabilitate s-ar putea să nu fie foarte entuziasmate în a dona capacitatea departamentului lor pentru a ajuta.

O a treia problemă este difuzarea. Cele mai multe reţele locale suportă difuzarea şi multe protocoale de nivel superior folosesc această facilitate în mod extensiv. Spre exemplu, atunci când un utilizator doreşte să trimită un pachet către o adresă IP x, cum ştie staţia sa ce adresă MAC să pună în cadru? Vom studia această întrebare în cap. 5, dar, pe scurt, răspunsul este că va difuza un cadru care conţine întrebarea: A cui este adresa IP x? Apoi aşteaptă un răspuns. Şi există multe alte exemple de utilizare a difuzării. Pe măsură ce din ce în ce mai multe reţele locale sunt interconectate, numărul cadrelor de difuzare recepţionate de fiecare maşină tinde să crească liniar cu numărul de maşini.

O altă problemă legată de difuzare apare din când în când, atunci când o placă de reţea se defectează şi începe să transmită un şir nesfârşit de cadre de difuzare. Rezultatul acestei **furtuni de difuzări** (broadcast storm) este că (1) întreaga capacitate a reţelei locale este ocupată de aceste cadre şi (2) că toate maşinile din toate reţele locale interconectate cu aceasta sunt paralizate doar prin procesarea şi ignorarea tuturor cadrelor difuzate.

La prima vedere s-ar putea părea că furtunile de difuzări pot fi limitate în spaţiu prin separarea reţelelor locale prin punţi şi comutatoare, dar dacă scopul este să se atingă transparenţă (de ex o maşină poate fi mutată intr-o reţea locală diferită fără ca nimeni să observe acest lucru), atunci punţile trebuie să înainteze toate cadrele de difuzare.

După ce am văzut de ce companiile ar dori să aibă mai multe reţele locale cu întindere limitată, haideţi să ne întoarcem la problema decuplării topologiei logice de cea fizică. Să presupunem că un utilizator este mutat în cadrul companiei de la un departament la altul fără să îşi schimbe biroul, sau că îşi schimbă biroul fără a-şi schimba departamentul. Folosind o cablare bazată pe hub-uri, mutarea utilizatorului în reţeaua locală corectă presupune ca administratorul de reţea să meargă în centrul de cablare şi să mute conectorul pentru calculatorul utilizatorului respectiv dintr-un hub în alt hub.

În multe companii, schimbările organizaţionale au loc tot timpul, însemnând că administratorii de sistem petrec o mulţime de timp scoţând cabluri de undeva şi punându-le în altă parte. De asemenea, în unele cazuri, este posibil ca schimbările să nu poată fi făcute deloc, pentru că perechea torsadată de la maşina utilizatorului este prea departe de hub-ul potrivit (de ex. în altă clădire).

Ca răspuns la cerinţele utilizatorilor pentru o flexibilitate sporită, comercianţii de echipamente de reţea au început să lucreze la o modalitate de a recabla clădiri în întregime doar cu ajutorul sof- tware-ului.. Conceptul rezultat este numit **VLAN (Virtual LAN**, rom: reţea locală virtuală) şi a fost standardizat de către comitetul 802. Acum este utilizat în multe organizaţii. Haideţi să aruncăm o privire asupra lui. Pentru informaţii suplimentare despre VLAN-uri, vezi (Breyer and Riley, 1999; and Seifert, 2000).

VLAN-urile se bazează pe comutatoare dedicate, cu toate că pot avea nişte hub-uri la periferie, ca în fig. 4-48. Pentru configurarea unei reţele bazate pe VLAN-uri, administratorul de reţea decide câte VLAN-uri vor exista, ce calculatoare vor aparţine fiecărui VLAN şi cum se vor numi VLAN- urile. De cele mai multe ori, VLAN-urile sunt denumite (informal) cu nume de culori, pentru că este apoi posibilă tip ărirea de diagrame color cu dispunerea fizică a maşinilor, figurând membrii VLAN-ului roşu în roşu, membrii VLAN-ului verde în verde şi aşa mai departe. În acest fel, atât dispunerea logică, cât şi cea fizică, sunt vizibile într-o singură figură.

Ca un exemplu, să considerăm cele patru reţele locale din fig. 4-49(a), în care opt dintre maşini aparţin VLAN-ului G (gri) şi şapte aparţin VLAN-ului A (alb). Cele patru reţele locale sunt conectate cu două punţi, B1 şi B2. Dacă este folosită cablare centralizată cu fire torsadate, pot fi de asemenea prezente 4 hub-uri (care nu sunt prezentate în figură), dar la nivel logic un cablu cu mai mulţi conectori şi un hub sunt acelaşi lucru. Prezentarea lor în modul în care sunt figuraţi aici face figura mai puţin încărcată. De asemenea, termenul de punte tinde să fie folosit în zilele noastre mai ales în cazurile când există mai multe maşini pe fiecare port, ca în această figură, dar în rest termenii “punte” şi “comutator” sunt interschimbabili. Fig. 4-49(b) prezintă aceleaşi maşini şi aceleaşi acelaşi VLAN-uri folosind comutatoare cu un singur calculator pe fiecare port.

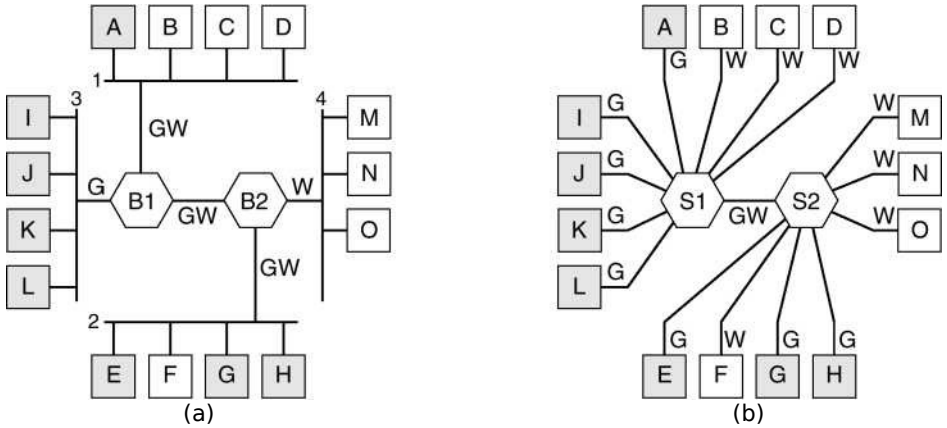


Fig. 4-49. (a) Patru reţele fizice organizate în două VLAN-uri, gri şi alb, de către două punţi
(b) Aceleaşi 15 maşini organizate în doua VLAN-uri cu comutatoare

Pentru a asigura funcţionarea corectă a VLAN-urilor, trebuie create tabele de configurare în comutatoare sau în punţi. Aceste tabele stabilesc care VLAN este accesibil pe fiecare dintre porturi (linii). Atunci când un cadru este recepţionat de la, să spunem, VLAN-ul gri, acesta trebuie înaintat către toate porturile marcate cu G. Acest lucru este valabil atât pentru traficul direcţionat, cât şi pentru cel cu destinaţie multiplă şi cu difuzare.

Observaţi faptul că un port poate fi marcat cu mai multe culori de VLAN-uri. Acest lucru poate fi văzut clar în fig. 4-49(a). Să presupunem că maşina A difuzează un cadru. Puntea B1 recepţionează cadrul şi observă că acesta este provenit de la o staţie din VLAN-ul gri, deci îl va înainta către toate porturile marcate cu G (cu excepţia portului de unde a venit). Din moment ce B1 are numai două alte porturi şi

ambele sunt marcate cu G, cadrul va fi trimis pe ambele porturi.

În cazul lui B2 povestea este diferită. Aici puntea ştie că nu există maşini gri în reţeaua locală 4, deci cadrul nu va fi înaintat acolo. Acesta va merge numai către reţeaua locală 2. Dacă unul dintre utilizatorii din reţeaua locală 4 îşi va schimba departamentul şi va fi mutat în VLAN-ul gri, atunci tabela din interiorul lui B2 va trebui actualizată pentru a reeticheta portul cu GA în loc de A. Dacă maşina F devine gri, atunci portul către reţeaua locală 2 trebuie etichetat cu G în loc de GA.

Acum să presupunem că toate maşinile atât din reţeaua locală 2 cât şi din reţeaua locală 4 devin gri. Atunci nu numai că porturile lui B2 către reţelele 2 şi 4 vor fi marcate cu G, dar şi portul lui B 1 către B2 trebuie de asemenea reetichetat de la GA la G, din moment ce cadrele albe care ajung la B1 din reţelele 1 şi 3 nu mai sunt înaintate către B2. În fig. 4-49(b) aceşti situaţie rămâne în picioare, numai că aici toate porturile care ajung la câte o singură maşină sunt etichetate cu o singură culoare deoarece acolo există un singur VLAN.

Până acum am presupus ca punţile şi comutatoarele ştiu cumva ce culoare are un cadru recepţionat. Cum ştiu acest lucru? Există trei metode folosite:

1. Fiecărui port îi este asociată o culoare de VLAN
2. Fiecărei adrese MAC îi este asociată o culoare de VLAN
3. Fiecărui protocol de nivel 3 sau fiecărei adrese IP îi este asociată o culoare de VLAN

Cu prima metodă, fiecare port este etichetat cu o culoare de VLAN. Totuşi, această metodă funcţionează doar dacă toate maşinile de pe un port aparţin aceluiaşi VLAN. În fig. 4-49(a), acest lucru este valabil în cazul lui B1 pentru portul către reţeaua 3, dar nu şi pentru portul către reţeaua 1.

În cazul celei de-a doua metode, puntea sau comutatorul are o singură tabelă ce conţine adresa MAC pe 48 de biţi a fiecărui maşini conectate la el, împreună cu VLAN-ul căruia îi aparţine maşina respectivă. În aceste condiţii, este posibilă combinarea mai multor VLAN-uri pe o singură reţea locală fizică, cum este cazul reţelei 1 din fig. 4-49(a). Când un cadru este recepţionat, tot ce trebuie să facă puntea sau comutatorul este să extragă adresa MAC şi să caute intrarea corespunzătoare din tabelă, pentru a găsi VLAN-ul de unde a fost recepţionat cadrul.

Cea de-a treia metodă presupune ca puntea sau comutatorul să examineze câmpul încărcare utilă al cadrului cu scopul de a clasifica, de exemplu, toate maşinile IP ca aparţinând unui VLAN şi toate maşinile AppleTalk ca aparţinând altuia. Pentru cel dintâi, adresa IP poate fi de asemenea utilizată pentru identificarea maşinii. Această strategie este foarte utilă atunci când mai oricare din mai multe maşini sau calculatoare portabile pot fi cuplate în mai multe staţii de ancorare. Din moment ce fiecare staţie de ancorare are propria adresă MAC, doar cunoaşterea staţiei de ancorare folosite nu spune nimic despre VLAN-ul căruia îi aparţine laptop-ul.

Singura problemă cu această abordare este că nu respectă una dintre regulile de bază în reţele ce calculatoare: independenţa nivelurilor. Nu este treaba nivelului legătură de date ce este în câmpul de încărcare utilă al cadrului. Acest nivel nu ar trebui să examineze aceste câmp şi cu atât mai puţin să ia decizii pe baza conţinutului acestuia. O consecinţă a utilizării acestei abordări este aceea că o modificare a unui

protocol de nivel 3 (de exemplu o trecere de la IPv4 la IPv6) va duce la nefuncţionarea comutatorului. Din nefericire, există pe piaţă comutatoare care funcţionează în acest fel.

Desigur, nu este nimic în neregulă în rutarea bazată pe adrese IP – aproape tot cap. 5 este dedicat rutării IP – dar să combini nivelurile înseamnă să o cauţi cu lumânarea. Un producător de comutatoare poate desconsidera acest argument susţinând că toate comutatoarele comercializate de el înţeleg atât IPv4, cât şi IPv6, deci totul este în regulă. Dar ce se va întâmpla atunci când va apărea IPv7? Producătorul probabil că va răspunde: cumpăraţi comutatoare noi, este asta atât de rău?

Standardul IEEE 802.1Q

Dacă ne gândim mai bine, ceea ce contează cu adevărat este VLAN-ul cadrului însuşi, nu VLAN-ul maşinii care l-a trimis. Dacă ar exista o modalitate de identificare a VLAN-ului în antetul cadrului, atunci necesitatea de a examina câmpul încărcare ar dispărea. Pentru un model nou de reţea locală, cum ar fi 802.11 sau 802.16, ar fi fost destul de uşor să fie adăugat numărul VLAN-ului în antet. De fapt, câmpul *identificator de conexiune* din 802.16 este oarecum similar cu spiritul identificatorilor de VLAN. Dar ce să facem cu Ethernetul, care este tehnologia dominantă de reţele locale şi care nu are câmpuri goale disponibile care să poată fi utilizate pentru identificatorul de VLAN?

Comitetul IEEE 802 a confruntat această problemă în 1995. După multe discuţii, a făcut inimaginabilul şi a modificat cadrul Ethernet. Noul format a fost publicat în standardul IEEE 802.1Q, lansat în 1998. Noul format conţine marcajul pentru VLAN; îl vom examina în curând. Nu în mod surprinzător, schimbarea a ceva atât de bine împământenit cum este Ethernetul nu este în întregime trivială. O serie de întrebări care ne vin în gând sunt:

1. Trebuie să aruncăm câteva sute de milioane de plăci de reţea Ethernet?
2. Dacă nu, cine generează noul câmp?
3. Ce se întâmplă cu cadrele care au deja lungimea maximă?

Desigur, comitetul 802 a fost conştient (în mod dureros) de aceste probleme şi a trebuit să ofere soluţii, ceea ce a şi făcut.

Cheia pentru găsirea soluţiei este să realizăm că identificatorii de VLAN sunt utilizaţi efectiv numai de punţi şi de comutatoare şi nu de către maşinile utilizatorilor. Prin urmare, în fig. 4-49 nu este esenţial ca identificatorii să fie prezenţi pe liniile ce pornesc de la staţii, atât timp cât sunt pre- zenţi pe liniile ce interconectează punţile. Prin urmare, pentru a folosi VLAN-uri, punţile şi comutatoarele trebuie să fie conştiente de existenţa acestora, dar aceasta era deja o cerinţă. Acum introducem necesitatea suplimentară ca acestea să implementeze 802.1Q, iar cele noi deja fac acest lucru.

La întrebarea dacă trebuie aruncate toate plăcile Ethernet, răspunsul este nu. Aduceţi-vă aminte: comisia 802.3 nu a putut convinge oamenii să schimbe câmpul tip intr-un alt câmp numit lungime. Va puteţi imagina reacţia acestora la anunţul ca toate plăcile Ethernet au fost scoase din uz. Oricum, în momentul în care noile placi Ethernet vor apărea pe piaţă, se spera că acestea vor suporta 802.1Q şi se vor putea

integra în totalitate în VLAN-uri.

Aşa că, daca cel care generează mesajul nu introduce câmpurile pentru VLAN, atunci cine o va face? Răspunsul este că prima punte sau comutator ce suporta VLAN la care ajunge un cadru, adaugă câmpurile, şi la ultimul le scoate. Dar cum ştie care cadru aparţine cărui VLAN ? Ei bine, prima punte sau primul comutator poate atribui un număr VLAN unui port, se poate uita la adresa MAC, sau să examineze informaţia utilă. Până când toate plăcile Ethernet vor în conformitate cu 802.1Q, suntem oarecum tot în punctul din care am plecat. Marea speranţă este că de la început toate plăcile gigabit Ethernet vor fi în conformitate cu 802.1Q şi pe măsură ce oamenii vor trece la gigabit Ethernet , 802.1Q va fi introdus automat. În ce priveşte problema cadrelor mai mari de 1518 octeţi, 802.1Q ridică limita la 1522 octeţi.

În timpul procesului de tranziţie, multe reţele vor avea ca maşini perimate (de obicei Ethernet clasic sau rapid) care nu suporta VLAN şi maşini (de obicei gigabit Ethernet) care suporta. Situaţia este arătată în fig. 4-50, unde simbolurile umbrite suportă VLAN iar celelalte nu. Pentru a simplifica problema, presupunem că toate comutatoarele suporta VLAN. Chiar dacă nu este cazul, primul comutator ce suportă VLAN poate adăuga marcaje bazate pe adrese MAC sau IP.

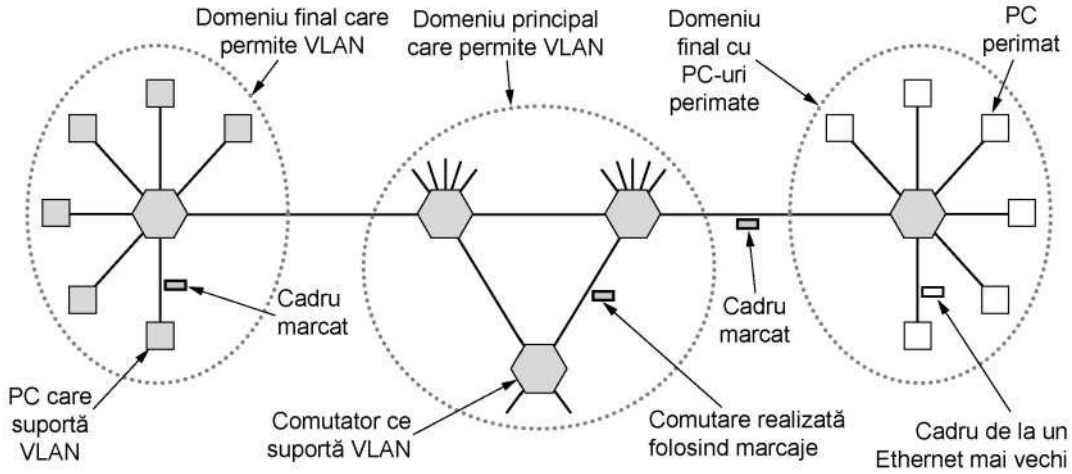

Domeniu final care permite VLAN
Domeniu principal care permite VLAN
Domeniu final cu PC-uri perimate
PC perimat
Cadru marcat
Cadru marcat
PC care suportă VLAN
Comutator ce suportă VLAN
Comutare realizată folosind marcaje
Cadru de la un Ethernet mai vechi

Fig. 4-50. Tranziţia de la un Ethernet perimat la un Ethernet ce suportă VLAN. Simbolurile umbrite suportă VLAN; cele goale nu.

În aceasta figură, plăcile Ethernet ce suportă VLAN generează direct cadre marcate (de exemplu 802.1Q), şi comutările ulterioare se folosesc de aceste marcaje. Pentru a face această comutare, comutatoarele trebuie sa ştie în prealabil care VLAN poate fi accesat şi pe ce port. Ştiind că un cadru aparţine unui VLAN gri , nu ajută prea mult faptul ca un cadru aparţine unui VLAN gri, până când comutatorul ştie care porturi sunt conectate la maşinile din VLAN-ul gri. Aşa că, comutatorul are nevoie de o tabela indexată de VLAN care să spună ce porturi sa folosească şi care suporta VLAN sau nu.

Când un PC perimat trimite un cadru către un comutator ce suporta VLAN, comutatorul construieşte un nou cadru marcat bazat pe cunoştinţele sale despre VLAN-ul care l-a trimis (folosind portul, adresa MAC sau adresa IP). Din acel punct,

nu mai contează dacă cel care trimite este o maşină legacy (perimată). Similar, un comutator care trebuie să trimită un cadru marcat către o maşină perimată (legacy) trebuie să reconstruiască cadrul în forma veche înainte de a-l furniza.

Haideţi să privim formatul cadrului 802.1Q. Este schiţat în fig. 4-51. Singura schimbare este adăugarea unei perechi de câmpuri a cate 2 octeţi. Primul este identificatorul protocolului VLAN. El are întotdeauna valoarea 0x8100. Întrucât acest număr este mai mare de 1500, toate plăcile Ethernet interpretează acest număr ca tip nu ca lungime. Ce face o placa mai veche cu un asemenea cadru este o problemă deoarece asemenea cadre nu ar trebui trimise către acestea.

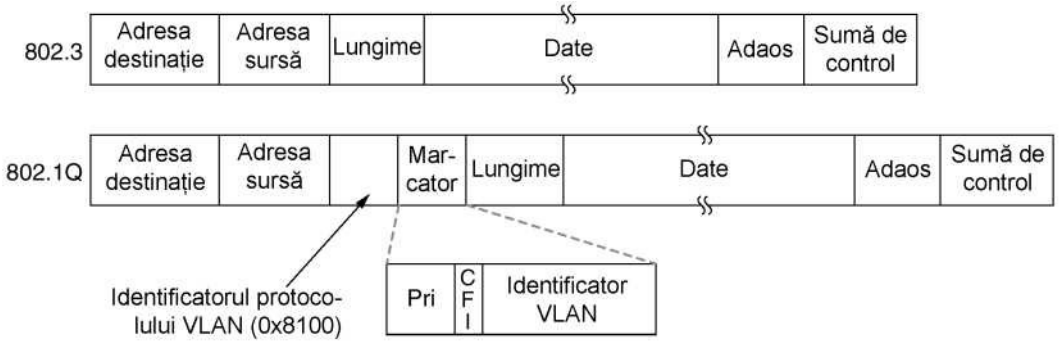


Fig. 4-51. Formatul cadrelor Ethernet 802.3 moştenite şi 802.1Q.

Al doilea câmp de 2 octeţi conţine trei sub-câmpuri. Sub-câmpul principal este identificatorul VLAN ce ocupa cei mai puţin semnificativi 12 octeţi. Aceasta este problema principala: cărui VLAN aparţine fiecare cadru? Câmpul Prioritate de 3 biţi nu are nici o legătură cu VLAN-ul, dar întrucât schimbarea antetului Ethernet este un eveniment foarte rar care s-ar desfăşura pe parcursul a trei ani şi ar implica 100 de oameni, de ce nu am pune alte informaţii folositoare în el? Acest câmp face posibila distingerea intre traficul în timp real implementat hard şi cel implementat soft şi de traficul intens pentru o mai buna calitate a serviciilor în Ethernet. Este nevoie de voce prin Ethernet (ca sa fim imparţiali, IP a avut un câmp similar mai mult de un sfert de secol şi nu a fost folosit niciodată).

Ultimul bit, CFI (Canonical Format Indicator, rom: indicator de format canonic) ar fi trebuit sa fie numit CEI (Corporate Ego Indicator, rom: indicator de ego al corporaţiei). Originar era folosit să indice adresele MAC în format little-indian sau big-indian însă această obişnuinţă s-a pierdut datorită controverselor. În zilele noastre prezenţa lui indică faptul că informaţiile utile conţin un 802.5 cadru prestabilit (freezed-dried, rom: îngheţat şi uscat) care este transportat de o reţea Ethernet care speră să găsească la destinaţie un LAN 802.5. Tot acest aranjament, nu are nici o legătura cu VLAN- urile. Însă politica comitetului de standarde este foarte asemănătoare cu politica obişnuită: dacă votezi în favoarea bitului meu, votez şi eu în favoarea bitului tău.

Aşa cum am precizat mai sus, când un cadru marcat ajunge la un comutator ce suporta VLAN, comutatorul foloseşte, ca un index intr-o tabela identificatorul VLAN, pentru a găsi la ce port sa trimită. Dar de unde vine tabela? Este construita manual, ne-am întors de unde am plecat: configurarea manuala a punţilor. Frumuseţea

punţilor transparente este faptul că acestea sunt montate şi pornite (plug-and-play) şi nu au nevoie de configurare manuală. Ar fi păcat să se piardă această facilitate. Din fericire, punţile care suportă VLAN se pot autoconfigura pe baza marcajelor care vin. Dacă un cadru marcat, cum ar fi VLAN 4, ajunge la portul 3, aparent câteva calculatoare de pe portul 3 aparţin VLAN-ului 4. Standardul 802.1Q explica cum să construieşti dinamic tabelele, în marea majoritate a cazurilor referindu-se la părţi apropiate din algoritmul lui Perlman standardizat în 802.1D.

Înainte de a părăsi subiectul referitor la rutarea în VLAN, merita să facem o ultima observaţie. Mulţi oameni în lumea Internetului şi a Ethernetului susţin fanatic reţelele neorientate pe conexiune şi se opun cu violenţă conexiunilor la nivelul legăturii de date. În momentul de faţă, VLAN-urile introduc ceva ce este surprinzător de asemănător cu o conexiune. Pentru a utiliza corect VLAN-uri, fiecare cadru are un nou identificator special care este utilizat pe post de index într-o tabelă din comutator, pentru a găsi destinaţia cadrului. Este exact acelaşi principiu de funcţionare ce apare la reţelele orientate pe conexiune. În reţelele neorientate pe conexiune adresa destinaţie este folosită la rutare şi nu exista identificatori de conexiune. Alte aspecte privind comunicarea vor fi tratate în cap. 5.

4.8 REZUMAT

Anumite reţele au un singur canal care este folosit pentru toate comunicaţiile. În aceste reţele, problema principală de proiectare este alocarea acestui canal între staţiile concurente care doresc să îl folosească. Au fost puşi la punct numeroşi algoritmi de alocare a canalului. Un rezumat al unora dintre cele mai importante metode de alocare a canalului este prezentat în fig. 4-52.

Metodă	Descriere
FDM	Dedică o bandă de frecvenţă fiecărei staţii
WDM	O schemă dinamică FDM pentru fibră optică
TDM	Dedică o cuantă de timp fiecărei staţii
ALOHA pur	Transmisie nesincronizată în orice moment
ALOHA cuantificat	Transmisie aleatoare în cuante de timp bine definite
CSMA l-persistent	Acces multiplu standard cu detectarea purtătoarei
CSMA nepersistent	Întârziere aleatoare când canalul este ocupat
CSMA p-persistent	CSMA cu probabilitatea de persistenţă p
CSMA/CD	CSMA cu oprire în cazul detectării unei coliziuni
Hartă de biţi (bit map)	Utilizează o hartă de biţi pentru planificare de tip rulare prin
Numărare binară inversă	Următoarea este staţia pregătită cu cel mai mare număr
Parcurgere arborescentă	Reduce conflictele prin activare selectivă
Divizarea lungimii de undă	Schemă FDM dinamică pentru fibre optice
MACA, MACAW	Protocoale LAN fără fir
Ethernet	CSMA/CD cu algoritm cu regresie exponenţială binară
FHSS	Frequency hopping spread spectrum
DSSS	Direct sequence spread spectrum
CSMA/CA	Acces multiplu cu sesizarea purtătoarei cu evitarea coliziunilor

Fig. 4-52. Metode şi sisteme de alocare a canalului pentru un canal obişnuit.

Cele mai simple scheme de alocare sunt FDM şi TDM. Acestea sunt eficiente atunci când numărul de staţii este mic şi traficul continuu, oarecum echilibrat. Amândouă sunt larg folosite în aceste condiţii, de exemplu pentru a diviza banda de legătură

utilizată pentru trunchiuri telefonice.

Dacă numărul staţiilor este mare şi variabil, iar traficul de tip rafală, atunci FDM şi TDM nu sunt alegeri bune. Ca alternativă a fost propus protocolul ALOHA, cu sau fără cuantificare şi control. ALOHA şi numeroasele sale variante şi derivate a fost pe larg discutat, analizat şi folosit în sisteme reale.

Atunci când starea canalului poate fi detectată, staţiile pot evita începerea unei transmisii cât timp transmite altă staţie. Această tehnică, detectarea purtătoarei, a condus la o diversitate de protocoale care pot fi folosite pe LAN-uri şi MAN-uri.

Există o clasă de protocoale care elimină total conflictele, sau cel puţin le reduce considerabil. Numărarea binară inversă elimină complet conflictele. Protocolul de parcurgere arborescentă le reduce împărţind dinamic staţiile în două grupuri disjuncte, unuia permiţându-i-se să transmită iar celuilalt nu. Acesta încearcă să facă împărţirea astfel, încât transmisia să-i fie permisă unei singure staţii dintre cele pregătite să trimită.

LAN-urile fără fir au propriile lor probleme şi soluţii. Cea mai mare problemă este cauzată de staţii ascunse, astfel încât CSMA nu funcţionează. O clasă de soluţii, tipizată de MACA şi MACAW, intenţionează să stimuleze transmisiile în jurul destinaţiei, pentru a îmbunătăţi funcţionarea CSMA.

FHSS şi DSSS sunt de asemenea utilizate. IEEE 802.11 combină CSMA şi MACAW pentru a produce CSMA/CA

Ethernetul este forma dominanta pentru reţele locale. Acesta utilizează CSMA/CD pentru alocarea canalului. Versiunile vechi foloseau cabluri ce şerpuiau de la o maşina la alta, dar acum sunt utilizate perechi de fire torsadate ce se conectează în noduri şi comutatoare. Vitezele au crescut de la 10 Mbps la 1 Gbps şi cresc în continuare.

LAN-urile fără fir sunt din ce în ce mai comune, 802.11 dominând acest domeniu. Nivelul său fizic permite cinci moduri de transmisie diferite, incluzând infraroşu, o varietate de spread spectrum schemes, şi un sistem FDM multicanal. Poate opera cu câte o staţie de bază în fiecare celulă, dar poate opera şi fără nici. Protocolul este o versiune de MACAW cu sesizarea virtuală a purtătoarei. MAN-urile fără fir au început deja să apară. Acestea sunt sisteme de bandă largă care utilizează unde radio pentru a înlocui ultimele porţiuni în conexiunile telefonice. Sunt folosite tehnici tradiţionale de modulaţie de bandă îngustă. Calitatea serviciilor este importantă, cu 802.16 definindu-se patru clase şi anume: viteză de transmisie constantă, două viteze de transmisie variabile şi o viteză de transmisie cu cea mai bună încercare (eng. best efforts).

Sistemul Bluetooth este de asemenea fără fir dar este adresat mai mult către sistemele desktop, pentru conectarea căştilor şi a altor echipamente la calculatoare fără a utiliza fire. Se intenţionează de asemenea conectarea perifericelor, cum ar fi faxurile la telefoane mobile. Ca şi 802.11, acesta foloseşte FHSS în banda ISM. Datorită nivelului de zgomot din multe medii şi datorită necesităţii unei interacţiuni în timp real, diferite protocoale înglobează mecanisme complicate pentru urmărirea şi corecţia erorilor.

Având atât de multe LAN-uri diferite, este necesară o metodă de a le interconecta. Punţile şi comutatoarele sunt folosite în acest scop. Algoritmul cu arbore de acoperire

este folosit pentru a construi punţi plug-and-play. O noua dezvoltare în domeniul interconectării LAN-urilor este VLAN, care separă topologia logică a LAN-urilor de topologia fizică. Un nou format pentru cadrele Ethernet (802.1Q) a fost introdus pentru a oferi o modalitate mai simplă de introducere a VLAN- urilor în organizaţii.

4.9 PROBLEME

1. Pentru această problema folosiţi o formula din acest capitol, însă înainte de a începe rezolvarea problemei scrieţi formula. Cadrele ajung aleator la un canal de 100 Mbps pentru transmitere. Dacă în momentul când un cadru ajunge avem canalul ocupat, acesta îşi aşteaptă rândul într-o coadă. Dimensiunea cadrului este distribuita exponential cu o medie de 10.000 biţi/cadru. Pentru fiecare din următoarele rate de sosire, precizaţi întârzierea medie a unui cadru, incluzând timpul cât acesta stă în coadă şi timpul cât durează transmisia.

a) 90 cadre/sec.
b) 900 cadre/sec.
c) 9000 cadre/sec.

Un grup de N staţii folosesc în comun un canal ALOHA pur de 56 Kbps. Fiecare staţie emite în medie un cadru de 1000 de biţi la fiecare 100 sec, chiar dacă cel precedent nu a fost încă trimis (de exemplu, staţiile folosesc zone tampon). Care este valoarea maximă a lui N?

Comparaţi întârzierea unui canal ALOHA pur cu aceea a unui canal ALOHA cuantificat la încărcare mică. Care dintre ele este mai mică? Motivaţi răspunsul.

Zece mii de staţii de rezervare a biletelor de avion concurează pentru folosirea unui singur canal ALOHA cuantificat. O staţie obişnuită face 18 cereri/oră. O cuantă este de 125 ps. Care este încărcarea totală aproximativă a canalului?

O populaţie mare de utilizatori ALOHA generează 50 cereri/sec, inclusiv originalele şi retransmisiile. Timpul este cuantificat în unităţi de 40 ms.

a) Care este şansa de succes a primei încercări?
b) Care este probabilitatea unui număr de exact k coliziuni urmate de un succes?
c) Câte încercări de transmisie ne aşteptăm să fie necesare?

Măsurătorile făcute asupra unui canal ALOHA cuantificat, cu un număr infinit de utilizatori, arată că 10% din cuante sunt nefolosite.

a) Care este încărcarea canalului, G?
b) Care este productivitatea?
c) Canalul este subîncărcat sau supraîncărcat?

Într-un sistem cuantificat ALOHA cu o populaţie infinită, numărul mediu de cuante pe care o statie le aşteaptă între o coliziune şi retransmisia ei, este 4. Reprezentaţi curba întârzierii în funcţie de productivitate, pentru acest sistem.

Cat timp o staţie s trebuie să aştepte în cel mai rău caz înainte de a putea transmite cadre intrun LAN ce foloseşte:

a) protocolul de bază harta de biţi?
b) protocolul lui Mok şi Ward cu permutare virtuală a numerelor staţiilor?

Un LAN foloseşte versiunea lui Mok şi Ward pentru numărătoarea inversă binară. La un anumit moment, cele zece staţii au numerele virtuale de staţie 8, 2, 4, 5, 1, 7, 3, 6, 9 şi 0. Următoarele trei staţii care trebuie să emită sunt 4, 3 şi 9, în această ordine. Care sunt noile numere virtuale de staţie după ce toate cele trei şi-au terminat transmisiile?

Şaisprezece staţii concurează pentru folosirea unui canal comun folosind protocolul cu parcurgere arborescentă adaptivă. Dacă toate staţiile ale căror adrese sunt numere prime devin brusc simultan disponibile, câte intervale de bit sunt necesare pentru a rezolva conflictul?

O colecţie de 2^n staţii folosesc protocolul cu parcurgere arborescentă adaptivă pentru a arbitra accesul la un cablu comun. La un moment dat, două dintre ele devin disponibile. Care este numărul minim, maxim şi mediu de cuante pentru a parcurge arborele dacă $2^n >> 1$?

LAN-urile fără fir pe care le-am studiat foloseau protocoale ca MACA în loc de CSMA/CD. În ce condiţii ar fi posibil să folosească CSMA/CD?

Care sunt caracteristicile comune ale protocoalelor de acces la canal WDMA şi GSM?

14. Şase staţii , de la A la F, comunica utilizând protocolul MACA. Este posibil ca doua transmisii sa aibă loc simultan? Explicaţi răspunsul.

15. O clădire cu 7 etaje are 15 birouri alăturate pe fiecare etaj. Fiecare birou conţine o priză de perete pentru un terminal pe peretele din faţă, astfel încât prizele formează o reţea rectangulară în plan vertical, cu o distanţă de 4 m între prize, atât pe orizontală cât şi pe verticală. Presupunând că este posibil să se monteze câte un cablu direct între orice pereche de prize, pe orizontală, verticală sau diagonală, câti metri de cablu sunt necesari pentru conectarea tuturor prizelor folosind:
a) O configuraţie stea cu un singur ruter în mijloc?
b) Un LAN 802.3?
c) O reţea de tip inel (fără fir central)?

16. Care este viteza (în bauds) a unui LAN 802.3 standard de 10 Mbps?

17. Schiţaţi codificarea Manchester pentru şirul de biţi: 0001110101.

18. Schiţaţi codificarea Manchester diferenţială pentru şirul de biţi din problema precedentă. Presupuneţi că linia este iniţial în stare jos.

19. Un LAN CSMA/CD de 10 Mbps (care nu e 802.3), lung de 1 km, are o viteză de propagare de 200 m/ps. Cadrele de date au o lungime de 256 biţi, incluzând 32 de biţi de antet, suma de control şi alte date suplimentare. Primul interval de bit după o transmitere efectuată cu succes este rezervat pentru receptor spre a ocupa canalul pentru a trimite un cadru de confirmare de 32 de biţi. Care este viteza efectivă de date, excluzând încărcarea suplimentară şi presupunând că nu sunt coliziuni?

20. Două staţii CSMA/CD încearcă să transmită fiecare fişiere mari (multicadru). După ce este trimis fiecare cadru, ele concurează pentru canal folosind algoritmul de regresie exponenţială binară. Care este probabilitatea terminării conflictului la runda k, şi care este numărul mediu de runde per conflict?

21. Să considerăm cazul unei reţele CSMA/CD de 1G bps, cu un cablu mai lung de 1 km, fără repetoare. Viteza semnalului pe cablu este de 200.000 km/s. Care este dimensiunea minimă a cadrului?

22. Un pachet IP ce trebuie transmis în Internet are 60 octeţi cu tot cu antete. Daca LLC nu este utilizat, este nevoie să se adauge informaţie de umplutură în cadrul Ethernet, şi dacă da, câţi octeţi?

23. Cadrele Ethernet trebuie să aibă o lungime minimă de 64 de octeţi pentru a avea siguranţa că emiţătorul încă mai emite, în cazul unei coliziuni la capatul celălalt al cablului. Fast Ethernet-ul are aceeaşi dimensiune minimă a cadrului de 64 de octeţi, dar poate emite biţii de zece ori mai rapid. Cum este posibil să se menţină aceeaşi dimensiune minimă a cadrului?

24. Autorii unor cărţi susţin că dimensiunea maximă a cadrului Ethernet este de 1518

octeţi în loc de 1500 octeţi. Au aceştia dreptate? Explicaţi răspunsul.

25. Specificaţiile 1000Base-SX spun că ceasul ar trebui să meargă la 1250 MHz, deşi Gigabit Ethernet ar trebui sa transmită 1 Gbps. Este folosit acest plus de viteză pentru a mări siguranţa transmisiei? Dacă nu, specificaţi ce se întâmplă.

26. Câte cadre pe secundă poate manevra gigabit Ethernet? Luaţi în considerare toate cazurile relevante. Sugestie: contează faptul că este o reţea gigabit Ethernet.

27. Numiţi două reţele care permit sa aibă cadre împachetate cap-la-cap. De ce se merita să ai această facilitate?

28. În fig. 4-27 sunt arătate patru staţii, A, B, C şi D. Care dintre ultimele doua staţii credeţi ca este mai aproape de A şi de ce?

29. Presupunând ca un 11-Mbps LAN 802.11b transmite cadre de 64-octeti cap-la-cap printr-un canal radio rata erorilor de 10^{-7} . Câte cadre pe secundă vor fi distruse în medie?

30. O reţea 802.16 are lungimea canalului de 20 MHz. Câţi biţi/sec pot fi transmişi la o staţie conectată?

31. IEEE 802.16 suportă patru clase de servicii. Care clasă este cea mai bună alegere pentru a transmite semnal video necomprimat?

32. Daţi două motive pentru care reţelele ar trebui să utilizeze corectarea erorilor în loc de detecţia erorilor şi retransmisia datelor?

33. În fig. 4-35, am văzut ca un dispozitiv Bluetooth poate fi în două piconet-uri în acelaşi timp. Există vreun motiv ca un dispozitiv sa nu fie stăpân în ambele piconet-uri în acelaşi timp?

34. Fig. 4-25 arată diferite protocoale de nivel fizic. Care dintre acestea este mai apropiat de protocolul de nivel fizic al Bluetooth? Care este marea diferenţă dintre cele doua?

35. Bluetooth suportă două tipuri de legătură între un stăpân şi un sclav. Care sunt acestea şi la ce sunt folosite fiecare?

36. Cadrul de semnalizare la FHSS (frequency hopping spread spectrum) varianta 802.11 conţine timpul de locuire (dwell time). Credeţi ca la Bluetooth, cadrul de semnalizare analog, conţine de asemenea timpul de locuire (dwell time)? Discutaţi răspunsul.

37. Consideraţi LAN-urile interconectate din fig. 4-44. Presupuneţi că gazda a şi b sunt în LAN-ul 1, c este în LAN-ul 2 şi d este în LAN-ul 8. Iniţial tabelele de dispersie din toate punţile sunt goale şi se foloseşte arborele de acoperire din fig. 4-44(b). Arătaţi cum tabelele de dispersie din punţi diferite se schimbă după fiecare din următoarele evenimente ce se succed : primul a, apoi b şi aşa mai departe.

a) a trimite către d.
b) c trimite către a.
c) d trimite către c.
d) d trimite către LAN-ul 6.
e) d trimite către a.

38. O consecinţă în folosirea unui arbore de acoperire pentru a retransmite cadre intr-un LAN extins este ca unele punţi nu participă la retransmiterea cadrelor. Identificaţi trei punţi de acest fel în fig. 4-44. Exista vreun motiv pentru a păstra aceste punţi, chiar daca ele nu sunt folosite pentru retransmitere?

39. Imaginaţi-vă ca un comutator are plăci de extensie pentru patru linii de intrare. Se întâmplă frecvent ca un cadru care ajunge pe una din aceste linii trebuie să iasă pe altă linie pe aceeaşi placă. Ce variante are proiectantul comutatorului pentru aceasta situaţie?

40. Un comutator proiectat pentru a fi utilizat cu un Ethernet rapid are un fund de sertar care poate transfera 10 Gbps. Câte cadre/sec poate manevra în cel mai rău caz?

41. Consideraţi reţeaua din fig. 4-49(a). Daca maşina J devine brusc albă; este nevoie de vreo schimbare la etichetare? Dacă da, ce anume?

42. Descrieţi pe scurt diferenţele dintre comutatoarele cu memorare şi retransmitere şi cele cu cut- through?

43. În ceea ce priveşte cadrele defecte, comutatoarele cu memorare şi retransmitere au un avantaj fata de cele cut-through. Explicaţi care sunt acestea.

44. Pentru a pune în funcţiune VLAN-uri, este nevoie de tabele de configuraţie în comutatoare şi punţi. Ce s-ar întâmpla daca VLAN-urile din fig. 4-49(a) ar utiliza noduri în loc de mediu partajat? Nodurile au nevoie de tabele de configurare? De ce sau de ce nu?

45. În fig. 4-50 comutatorul din domeniul final cu PC îmbătrânite, figurat în dreapta este un comutator pregătit pentru VLAN. Este posibilă utilizarea unui comutator vechi în acest caz? Dacă da, cum va funcţiona acesta? Dacă nu, de ce?

46. Scrieţi un program care să simuleze comportamentul protocolului CSMA/CD în Ethernet când exista N staţii pregătite sa transmită în timp ce se transmite un cadru. Programul vostru trebuie să prezinte timpii când fiecare staţie începe sa transmită cu succes cadrul. Presupuneţi că un tact de ceas apare odată la fiecare cuantă de timp (51,2 microsecunde) şi o detecţie de coliziune şi o secvenţă de bruiaj durează o cuantă de timp. Toate cadrele sunt de dimensiune maximă admisă.

NIVELUL REŢEA

Nivelul reţea are ca sarcină preluarea pachetelor de la sursă şi transferul lor către destinaţie. Ajungerea la destinaţie poate necesita mai multe salturi prin rutere intermediare de-a lungul drumului. Această funcţie contrastează clar cu cea a nivelului legătură de date, care avea scopul mult mai modest de a transfera cadre de la un capăt al unui fir la celălalt. Astfel nivelul reţea este cel mai scăzut nivel care se ocupă de transmisii capăt la capăt.

Pentru realizarea scopurilor propuse, nivelul reţea trebuie să cunoască topologia subreţelei de comunicaţie (de exemplu mulţimea tuturor ruterelor) şi să aleagă calea cea mai potrivită prin aceasta. De asemenea trebuie să aleagă căile de urmat astfel, încât să nu încarce excesiv unele legături de comunicaţie sau rutere în timp ce altele sunt inactive. În fine, când sursa şi destinaţia fac parte din reţele diferite, apar probleme noi. Este sarcina nivelului reţea să se ocupe de ele. În acest capitol vom studia toate aceste aspecte şi le vom exemplifica, în primul rând folosind Internetul şi protocolul lui la nivelul reţea, IP, cu toate că vom vorbi şi despre reţele fără fir.

5.1 CERINŢELE DE PROIECTARE ALE NIVE LULUI REŢEA

Vom prezenta, în continuare, o introducere a cerinţelor pe care proiectantul nivelului reţea trebuie să le rezolve. Acestea includ serviciile furnizate nivelului transport şi proiectarea internă a subreţelei.

5.1.1 Comutare de pachete de tip Memorează-şi-Retransmite (Store-and-Forward)

Dar înainte de a începe explicarea detaliilor nivelului reţea, merită probabil sa reiniţializăm contextul în care operează protocoalele de la nivelul reţea. Acest context este prezentat în fig. 5-1. Componentele majore ale sistemului sunt echipamentul companiei de telecomunicaţii (rutere conectate prin linii de transmisie), prezentat în interiorul ovalului umbrit, şi echipamentul clientului, prezentat în afara ovalului. Gazda *H1* este conectată direct la unul dintre ruterele companiei de telecomunicaţii, *A,* printr-o linie închiriată. În contrast, *H2* este într-o reţea LAN cu un ruter, *F,* deţinut şi operat de către client. Acest ruter are, deasemeni, şi o linie închiriată către echipamentul companiei de telecomunicaţii. Am prezentat *F* ca fiind în afara ovalului, deoarece nu aparţine companiei de telecomunicaţii, dar în termeni de construcţie, software şi protocoale, probabil că nu diferă faţă de ruterele acesteia. Este discutabil dacă aparţine subreţelei, dar în contextul acestui capitol ruterele din localul clientului sunt considerate parte a subreţelei deoarece rulează aceeaşi algoritmi ca şi ruterele companiei de telecomunicaţii (şi aici principala noastră preocupare sunt algoritmii).

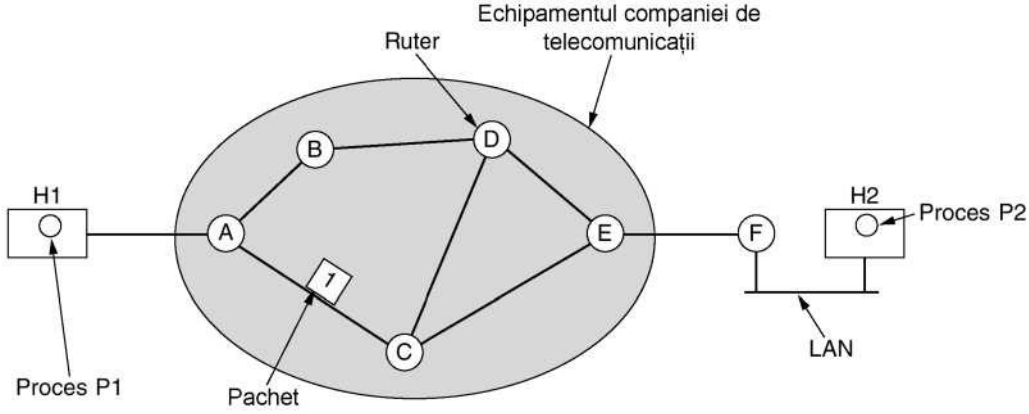


Fig. 5-1. Cadrul protocoalelor nivelului reţea.

Acest echipament este folosit după cum urmează. O gazdă care are de transmis

un pachet îl transmite celui mai apropiat ruter, fie în aceeaşi reţea LAN, fie printr-o legătură punct la punct cu compania de telecomunicaţii. Pachetul este memorat acolo până ajunge integral, astfel încât să poată fi verificată suma de control. Apoi este trimis mai departe către următorul ruter de pe traseu, până ajunge la gazda destinaţie, unde este livrat. Acest mecanism reprezintă comutarea de pachete de tip memorează-şi-retransmite, aşa cum am văzut în capitolele anterioare.

5.1.2 Servicii furnizate nivelului transport

Nivelul reţea furnizează servicii nivelului transport la interfaţa dintre cele două niveluri. O întrebare importantă este ce fel de servicii furnizează nivelul reţea nivelului transport. Serviciile nivelului reţea au fost proiectate având în vedere următoarele scopuri:

1. Serviciile trebuie să fie independente de tehnologia ruterului.
2. Nivelul transport trebuie să fie independent de numărul, tipul şi topologia ruterelor existente.
3. Adresele de reţea disponibile la nivelul transport trebuie să folosească o schemă de numerotare uniformă, chiar în cadrul reţelelor LAN şi WAN.

Obiectivele fiind stabilite, proiectantul nivelului reţea are o mare libertate în a scrie specificaţiile detaliate ale serviciilor oferite nivelului transport. Această libertate degenerează adesea într-o aprigă bătălie între două tabere opuse. Problema centrală a discuţiei este dacă nivelul reţea trebuie să furnizeze servicii orientate pe conexiune sau servicii neorientate pe conexiune.

O tabără (reprezentată de comunitatea Internet) afirmă că scopul ruterului este de a transfera pachete şi nimic mai mult. În viziunea lor (bazată pe experienţa a aproape 30 de ani de exploatare a unei reţele de calculatoare în funcţiune), subreţeaua este inerent nesigură, indiferent cum ar fi proiectată. De aceea calculatoarele gazdă trebuie să accepte faptul că reţeaua este nesigură şi să facă controlul erorilor (i.e., detecţia şi corecţia erorii) şi controlul fluxului ele însele.

Acest punct de vedere duce rapid la concluzia că serviciul reţea trebuie să fie neorientat pe conexiune, cu două primitive SEND PACKET şi RECEIVE PACKET şi cu foarte puţin în plus. În particular, nu trebuie făcută nici o operaţie pentru controlul ordinii sau fluxului pachetelor pentru că oricum calculatorul gazdă va face acest lucru, şi, de obicei, dublarea acestor operaţii aduce un câştig nesemnificativ. În continuare, fiecare pachet va trebui să poarte întreaga adresă de destinaţie, pentru că fiecare pachet este independent de pachetele predecesoare, dacă acestea există.

Cealaltă tabără (reprezentată de companiile de telefoane) afirmă că subreţeaua trebuie să asigure un serviciu orientat pe conexiune sigur. Ei susţin că 100 de ani de experienţă cu sistemul telefonic mondial reprezintă un ghid excelent. În această perspectivă, calitatea serviciului este elementul dominant, şi într-o subreţea fără conexiuni, calitatea serviciului este dificil de obţinut, în special pentru trafic în timp real cum ar fi voce şi imagine.

Aceste două tabere sunt cel mai bine exemplificate de Internet şi reţele ATM.

Reţeaua Internet oferă un serviciu la nivelul reţea neorientat pe conexiune; reţelele ATM oferă un serviciu la nivelul reţea orientat pe conexiune. Totuşi, este interesant de notat că cu cât garantarea calităţii serviciului devine din ce în ce mai importantă, Internetul evoluează. În particular, începe să dobândească proprietăţi asociate normal cu serviciile orientate conexiune, aşa cum vom vedea mai târziu. De fapt, ne-am făcut o părere despre această evoluţie în timpul studiului despre reţele VLAN în Cap. 4.

5.1.3 Implementarea serviciului neorientat pe conexiune

După ce am văzut cele două clase de servicii pe care nivelul reţea le furnizează utilizatorilor săi, este momentul să vedem funcţionarea internă a acestui nivel. Sunt posibile două organizări diferite, în funcţie de tipul serviciului oferit. Dacă este oferit un serviciu neorientat pe conexiune, atunci pachetele sunt trimise în subreţea individual şi dirijate independent de celelalte. Nu este necesară nici o iniţializare prealabilă. În acest context, pachetele sunt numite frecvent **datagrame** (datagrams) (prin analogie cu telegramele), iar subreţeaua este numită **subreţea datagramă** (datagram subnet). Dacă este folosit serviciul orientat conexiune, atunci, înainte de a trimite pachete de date, trebuie stabilită o cale de la ruterul sursă la ruterul destinaţie. Această conexiune este numită **VC** (**virtual circuit**, circuit virtual), prin analogie cu circuitele fizice care se stabilesc în sistemul telefonic, iar subreţeaua este numită **subreţea cu circuite virtuale** (**virtual-circuit subnet**). În această secţiune vom studia sub- reţele datagramă; în următoarea secţiune vom studia subreţelele cu circuite virtuale.

Să vedem cum funcţionează o subreţea datagramă. Să presupunem că procesul *P1* din fig. 5-2 are un mesaj lung pentru procesul *P2.* El transmite mesajul nivelului transport, cu instrucţiunile de livrare către procesul *P2* aflat pe calculatorul gazdă *H2.* Codul nivelului transport rulează pe calculatorul

gazdă *H1,* de obicei în cadrul sistemului de operare. Acesta inserează la începutul mesajului un antet corespunzător nivelului transport şi transferă rezultatul nivelului reţea, probabil o altă procedură din cadrul sistemului de operare.

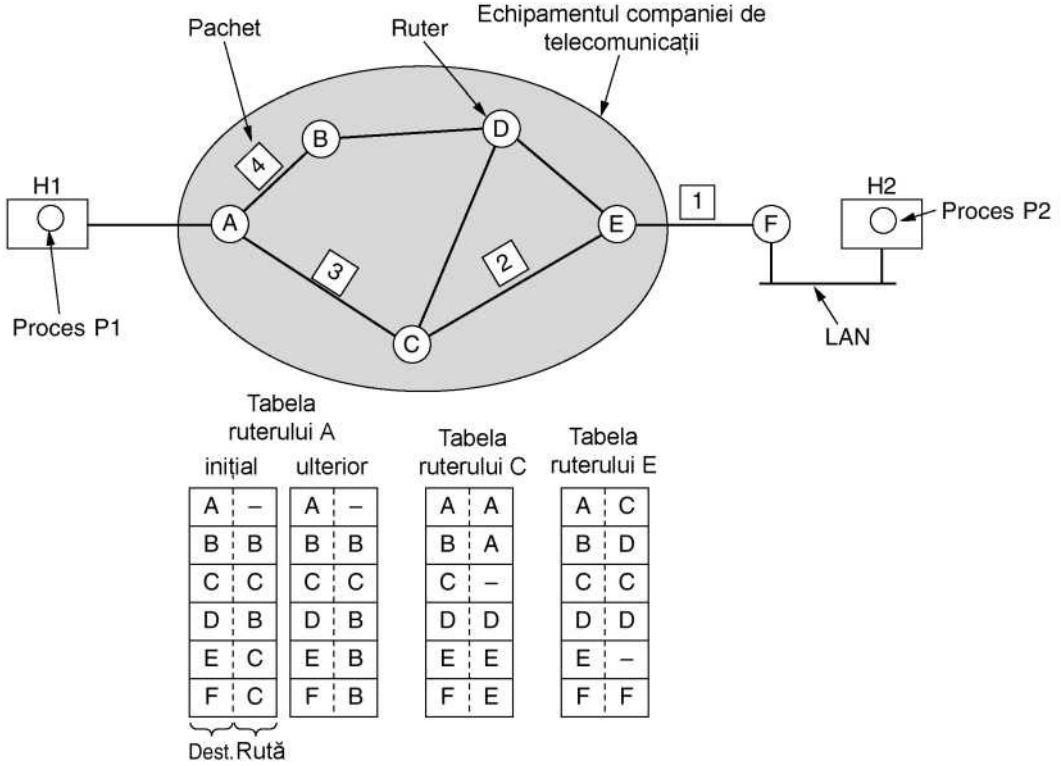


Fig. 5-2. Dirijarea într-o subreţea datagramă.

Să presupunem că mesajul este de patru ori mai lung decât dimensiunea maximă a unui pachet, aşa că nivelul reţea trebuie să îl spargă în patru pachete, 1, 2, 3, şi 4 şi să le trimită pe fiecare în parte ruterului *A,* folosind un protocol punct-la-punct, de exemplu, PPP. Din acest punct controlul este preluat de compania de telecomunicaţii. Fiecare ruter are o tabelă internă care îi spune unde să trimită pachete pentru fiecare destinaţie posibilă. Fiecare intrare în tabelă este o pereche compusă din destinaţie şi linia de ieşire folosită pentru acea destinaţie. Pot fi folosite doar linii conectate direct. De exemplu, în fig. 5-2, *A* are doar două linii de ieşire – către *B* şi *C* – astfel că fiecare pachet ce vine trebuie trimis către unul dintre aceste rutere, chiar dacă ultima destinaţie este alt ruter. Tabela de rutare iniţială a lui *A* este prezentată în figură sub eticheta „iniţial”.

Cum au ajuns la *A*, pachetele 1, 2 şi 3 au fost memorate pentru scurt timp (pentru verificarea sumei de control). Apoi fiecare a fost trimis mai departe către *C* conform tabelei lui *A*. Pachetul 1 a fost apoi trimis mai departe către *E* şi apoi către *F.* Când a ajuns la *F,* a fost încapsulat într-un cadru al nivelului legătură de date şi trimis către calculatorul gazdă *H2* prin reţeaua LAN.

Totuşi, ceva diferit s-a întâmplat cu pachetul 4. Când a ajuns la *A* a fost trimis

către ruterul *B,* chiar dacă şi el este destinat tot lui *F*. Dintr-un motiv oarecare, *A* a decis să trimită pachetul 4 pe o rută diferită de cea urmată de primele trei. Poate că a aflat despre o congestie undeva pe calea *ACE* şi şi-a actualizat tabela de rutare, aşa cum apare sub eticheta „mai târziu”. Algoritmul ce administrează tabelele şi ia deciziile de rutare se numeşte **algoritm de rutare** (routing algorithm). Algoritmii de rutare sunt unele dintre principalele elemente pe care le vom studia în acest capitol.

5.1.4 Implementarea serviciilor orientate pe conexiune

Pentru serviciile orientate conexiune, avem nevoie de o subreţea cu circuite virtuale. Să vedem cum funcţionează aceasta. Ideea care se stă la baza circuitelor virtuale este evitarea alegerii unei noi căi (rute) pentru fiecare pachet trimis, ca în fig. 5-2. În schimb, atunci când se stabileşte o conexiune, se alege o cale între maşina sursă şi maşina destinaţie, ca parte componentă a iniţializării conexiunii şi aceasta este memorată în tabelele ruterelor. Acea cale este folosită pentru tot traficul de pe conexiune, exact în acelaşi mod în care funcţionează sistemul telefonic. Atunci când conexiunea este eliberată, este închis şi circuitul virtual. În cazul serviciilor orientate conexiune, fiecare pachet poartă un identificator care spune cărui circuit virtual îi aparţine.

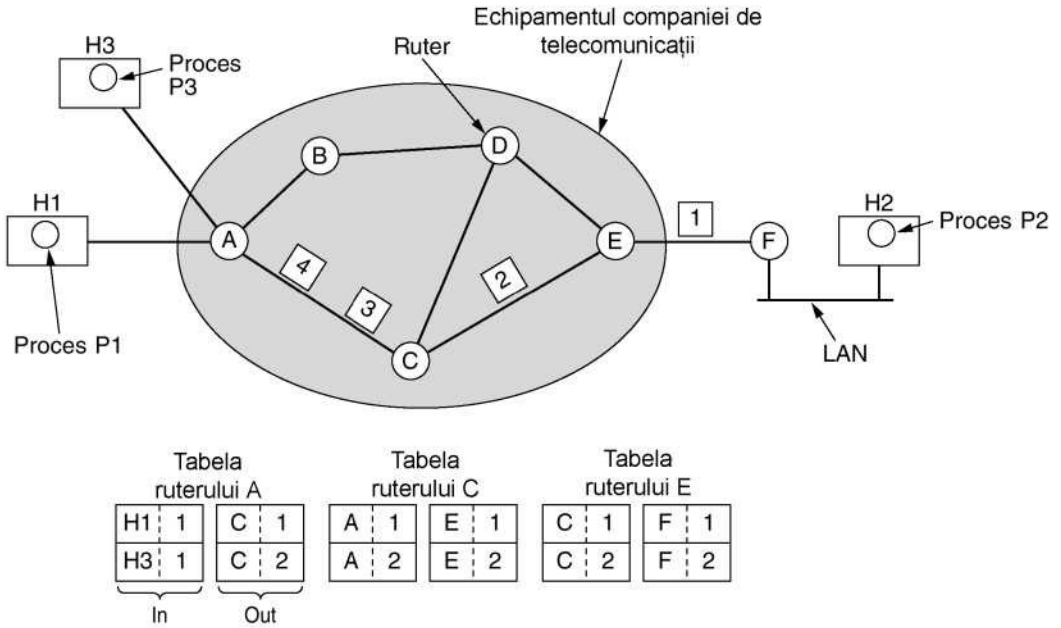


Fig. 5-3. Dirijare în cadrul unei subreţele cu circuite virtuale.

De exemplu, să considerăm situaţia din fig. 5-3. Aici calculatorul gazdă *H1* a stabilit conexiunea 1 cu calculatorul gazdă *H2.* Aceasta este memorată ca prima intrare în fiecare tabelă de rutare. Prima linie a tabelei lui *A* spune că dacă un pachet purtând identificatorul de conexiune 1 vine de la *H1*, atunci trebuie trimis

către ruterul *C,* dându-i-se identificatorul de conexiune 1. Similar, prima intrare a lui C dirijează pachetul către *E,* tot cu identificatorul de conexiune 1.

Acum să vedem ce se întâmplă dacă *H3* vrea, de asemenea, să stabilească o conexiune cu *H2*. Alege identificatorul de conexiune 1 (deoarece iniţializează conexiunea şi aceasta este singura conexiune) şi indică subreţelei să stabilească circuitul virtual. Aceasta conduce la a doua linie din tabele. Observaţi că apare un conflict deoarece deşi *A* poate distinge uşor pachetele conexiunii 1 de la *H1* de pachetele conexiunii 1 de la *H3, C* nu poate face asta. Din acest motiv, *A* asociază un identificator de conexiune diferit pentru traficul de ieşire al celei de a doua conexiuni. Pentru evitarea conflictelor de acest gen ruterele trebuie să poată înlocui identificatorii de conexiune în pachetele care pleacă. În unele contexte, aceasta se numeşte comutarea etichetelor (label switching).

5.1.5 Comparaţie între subretele cu circuite virtuale şi subretele datagramă

Atât circuitele virtuale cât şi datagramele au suporteri şi oponenţi. Vom încerca acum să rezumăm argumentele ambelor tabere. Principalele aspecte sunt prezentate în fig. 5-4, deşi cei extrem de riguroşi ar putea probabil găsi un contraexemplu pentru toate cele descrise în această figură.

Problemă	**Subreţea datagramă**	**Subreţea cu circuite virtuale (CV)**
Stabilirea circuitului	Nu este necesară	Obligatorie
Adresare	Fiecare pachet conţine adresa completă pentru sursă şi	Fiecare pachet conţine un număr mic de CV '
Informaţii de stare	Ruterele nu păstrează informaţii despre conexiuni	Fiecare CV necesită spaţiu pentru tabela ruterului per conexiune
Dirijare	Fiecare pachet este dirijat independent	Calea este stabilită la iniţierea CV; toate pachetele o urmează
Efectul defectării ruterului	Nici unul, cu excepţia pachetelor pierdute în timpul	Toate circuitele virtuale care trec prin ruterul defect sunt terminate
Calitatea serviciului	Dificil	Simplu, dacă pentru fiecare CV pot fi alocate în avans suficiente resurse
Controlul congestiei	Dificil	Simplu, dacă pentru fiecare CV pot fi alocate în avans suficiente resurse

Fig. 5-4. Comparaţie între subreţele datagramă şi subreţele cu circuite virtuale.

În interiorul subreţelei există situaţii în care trebuie să se aleagă între facilităţi antagoniste specifice fie circuitelor virtuale, fie datagramelor. Un astfel de compromis este acela între spaţiul de memorie al ruterului şi lăţimea de bandă. Circuitele virtuale permit pachetelor să conţină numere de circuite în locul unor adrese complete. Dacă pachetul tinde să fie foarte mic, atunci existenţa unei adrese complete în fiecare pachet poate reprezenta o supraîncărcare (overhead) importantă şi deci o irosire a lăţimii de bandă. Preţul plătit pentru folosirea internă a circuitelor virtuale este spaţiul necesar păstrării tabelei în ruter. Soluţia mai ieftină este determinată de raportul între costul circuitelor de comunicaţie şi cel al memoriei ruterului.

Alt compromis este cel între timpul necesar stabilirii circuitului şi timpul de

analiză a adresei. Folosirea circuitelor virtuale presupune existenţa unei faze iniţiale de stabilire a căii, care cere timp şi consumă resurse. Oricum, este uşor să ne imaginăm ce se întâmplă cu un pachet de date într-o sub- reţea bazată pe circuite virtuale: ruterul foloseşte numărul circuitului ca un index într-o tabelă pentru a afla unde merge pachetul. Într-o reţea bazată pe datagrame, pentru a găsi intrarea corespunzătoare destinaţiei se foloseşte o procedură de căutare mult mai complicată.

O altă problemă este cea a dimensiunii spaţiului necesar pentru tabela din memoria ruteru- lui. O subreţea datagramă necesită o intrare pentru fiecare destinaţie posibilă, în timp ce o reţea cu circuite virtuale necesită o intrare pentru fiecare circuit virtual. Totuşi, acest avantaj este relativ iluzoriu deoarece şi pachetele de iniţializare a conexiunii trebuie rutate, iar ele folosesc adresele destinaţie, la fel ca şi datagramele.

Circuitele virtuale au unele avantaje în garantarea calităţii serviciului şi evitarea congestionării subreţelei, deoarece resursele (de exemplu zone tampon, lărgime de bandă şi cicluri CPU) pot fi rezervate în avans, atunci când se stabileşte conexiunea. La sosirea pachetelor, lăţimea de bandă necesară şi capacitatea ruterului vor fi deja pregătite. Pentru o subreţea bazată pe datagrame, evitarea congestionării este mult mai dificilă.

Pentru sistemele de prelucrare a tranzacţiilor (de exemplu apelurile magazinelor pentru a verifica cumpărături realizate cu cărţi de credit) overhead-ul implicat de stabilirea şi eliberarea unui circuit virtual poate reduce cu uşurinţă utilitatea circuitului. Dacă majoritatea traficului este de acest tip, folosirea internă a circuitelor virtuale în cadrul subreţelei nu prea are sens. Pe de altă parte, ar putea fi de folos circuite virtuale permanente, stabilite manual şi care să dureze luni sau chiar ani.

Circuitele virtuale au o problemă de vulnerabilitate. Dacă un ruter se defectează şi îşi pierde conţinutul memoriei, atunci toate circuitele virtuale care treceau prin el sunt suprimate, chiar dacă acesta îşi revine după o secundă. Prin contrast, dacă se defectează un ruter bazat pe datagrame vor fi afectaţi doar acei utilizatori care aveau pachete memorate temporar în cozile de aşteptare ale ruterului şi este posibil ca numărul lor să fie şi mai mic, în funcţie de câte pachete au fost deja confirmate. Pierderea liniei de comunicaţie este fatală pentru circuitele virtuale care o folosesc, însă poate fi uşor compensată dacă se folosesc datagrame. De asemenea, datagramele permit ruterului să echilibreze traficul prin subreţea, deoarece căile pot fi modificate parţial în cursul unei secvenţe lungi de pachete transmise.

5.2 ALGORITMI DE DIRIJARE

Principala funcţie a nivelului reţea este dirijarea pachetelor de la maşina sursă către maşina destinaţie. În majoritatea subreţelelor pachetele vor face salturi multiple pentru a ajunge la destinaţie. Singura excepţie remarcabilă o reprezintă

reţelele cu difuzare, dar chiar şi aici dirijarea este importantă, atunci când sursa şi destinaţia nu sunt în aceeaşi reţea. Algoritmii care aleg calea şi structurile de date folosite de aceştia reprezintă un domeniu important al proiectării nivelului reţea.

Algoritmul de dirijare (routing algorithm) este acea parte a software-ului nivelului reţea care răspunde de alegerea liniei de ieşire pe care trebuie trimis un pachet recepţionat. Dacă subreţeaua foloseşte intern datagrame, această decizie trebuie luată din nou pentru fiecare pachet recepţionat, deoarece este posibil ca cea mai bună rută să se fi modificat între timp. Dacă subreţeaua foloseşte circuite virtuale, deciziile de dirijare sunt luate doar la iniţializarea unui nou circuit virtual. După aceea pachetele de date vor urma doar calea stabilită anterior. Acest ultim caz este numit uneori **dirijare de sesiune** (**session routing**)**,** deoarece calea rămâne în funcţiune pentru o întreagă sesiune utilizator (de exemplu o sesiune de conectare de la un terminal -login- sau un transfer de fişiere).

Uneori este util să se facă distincţia între dirijare, care înseamnă alegerea căii care va fi folosită, şi retransmitere, care se referă la ceea ce se întâmplă atunci când soseşte un pachet. Se poate spune despre un ruter că rulează intern două procese. Unul dintre ele preia fiecare pachet care soseşte, căutând în tabela de dirijare linia de ieşire folosită pentru el. Acesta este procesul de **retransmitere** (**forwarding**). Celălalt proces se ocupă de completarea şi actualizarea tabelei de rutare. Aici algoritmul intervine de dirijare.

Indiferent dacă ruta se alege independent pentru fiecare pachet sau doar la stabilirea unei noi conexiuni, un algoritm de dirijare trebuie să aibă anumite proprietăţi: corectitudine, simplitate, robusteţe, stabilitate, echitate, optimalitate. Corectitudinea şi simplitatea nu mai au nevoie de comentarii, dar necesitatea robusteţii poate fi mai puţin evidentă la prima vedere. Odată ce apare pe piaţă o reţea importantă, este de aşteptat ca ea să funcţioneze continuu ani întregi, fără defecte generale ale sistemului. În acest timp vor exista defecte hardware şi software de tot felul. Calculatoare gazdă, rutere, linii de comunicaţie vor cădea repetat şi topologia se va schimba de multe ori. Algoritmul de dirijare trebuie să facă faţă acestor modificări ale topologiei şi traficului, fără a impune ca toate job-urile de pe toate calculatoarele să fie abandonate şi reţeaua să fie reiniţializată de fiecare dată când se defectează un ruter.

Stabilitatea este de asemenea un obiectiv important pentru algoritmul de dirijare. Există algoritmi de dirijare care niciodată nu converg la echilibru, indiferent cât timp ar rula. Un algoritm stabil atinge starea de echilibru şi o menţine. Echitatea şi optimalitatea sunt evidente – este sigur că nici o persoană înţeleaptă nu li se opune – însă, aşa cum se va arăta, adeseori acestea sunt obiective contradictorii. Un exemplu simplu al acestui conflict este prezentat în fig. 5-5. Presupunem că între A şi A', între B şi B' şi între C şi C' există un trafic suficient pentru a satura legăturile orizontale. Pentru a maximiza fluxul total, traficul între X şi X' trebuie oprit. Din păcate acest lucru ar defavoriza pe X şi X'. Evident, este necesar un compromis între eficienţa globală şi echitatea faţă de fiecare dintre conexiuni.

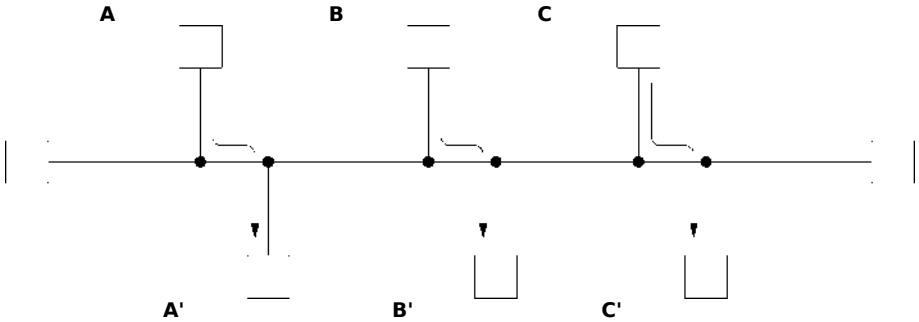


Fig. 5-5. Conflict între echitate şi optimalitate.

Înainte de a încerca să găsim rezolvarea acestui conflict între optimalitate şi prevenirea defavorizării, trebuie să stabilim ce vrem să optimizăm. Minimizarea întârzierii medii a unui pachet este un candidat evident, însă la fel este şi maximizarea productivităţii (throughput) totale a reţelei. Mai mult, şi aceste două obiective sunt în conflict, deoarece funcţionarea unui sistem cu cozi de aşteptare la limita capacităţii sale produce întârzieri majore. Pentru a realiza un compromis, în multe reţele se încearcă minimizarea numărului de salturi pe care trebuie să le facă un pachet, deoarece reducerea numărului de salturi tinde să îmbunătăţească întârzierea şi de asemenea să reducă lăţimea de bandă consumată, ceea ce tinde să îmbunătăţească şi productivitatea.

Algoritmii de dirijare pot fi grupaţi în două mari clase: neadaptivi şi adaptivi. **Algoritmii neadaptivi** (**nonadaptive algorithms**) nu îşi bazează deciziile de dirijare pe măsurători sau estimări ale traficului şi topologiei curente. Astfel, alegerea căii folosite pentru a ajunge de la nodul *I* la nodul *J* (oricare ar fi *I* şi *J)* se calculează în avans, off-line şi parvine ruterului la iniţializarea reţelei. Această procedură se mai numeşte şi **dirijare statică** (**static routing**).

Algoritmii adaptivi (**adaptive algorithms**), prin contrast, îşi modifică deciziile de dirijare pentru a reflecta modificările de topologie şi de multe ori şi pe cele de trafic. Algoritmii adaptivi diferă prin locul de unde îşi iau informaţia (de exemplu local, de la un ruter vecin sau de la toate ruterele), prin momentul la care schimbă rutele (de exemplu la fiecare AT secunde, când se schimbă încărcarea sau când se schimbă topologia) şi prin metrica folosită pentru optimizare (de exemplu distanţa, numărul de salturi sau timpul estimat pentru tranzit). În secţiunile următoare vom discuta o varietate de algoritmi de dirijare, atât statici cât şi dinamici.

5.2.1 Principiul optimalităţii

Înainte de a intra în algoritmii specifici, ar fi poate folositor să observăm că se poate face o afirmaţie despre rutele optimale fără a ne referi la topologia reţelei sau la trafic. Această afirmaţie este cunoscută sub numele de **principiul optimalităţii** (**optimality principle**). El stabileşte că dacă ruterul *J* este pe calea optimă de la ruterul *I* către ruterul *K,* atunci calea optimă de la *J* la *K* este pe aceeaşi rută. Pentru a vedea aceasta, să notăm cu r_1 partea din cale de la *I* la *J,* iar cu r_2 restul rutei. Dacă ar exista o rută mai bună decât r_2 de la *J* la *K,* ea ar putea fi concatenată cu r_1 şi ar îmbunătăţi ruta de la *I* la *K,* ceea ce ar contrazice presupunerea că $r_1 r_2$ este optimală.

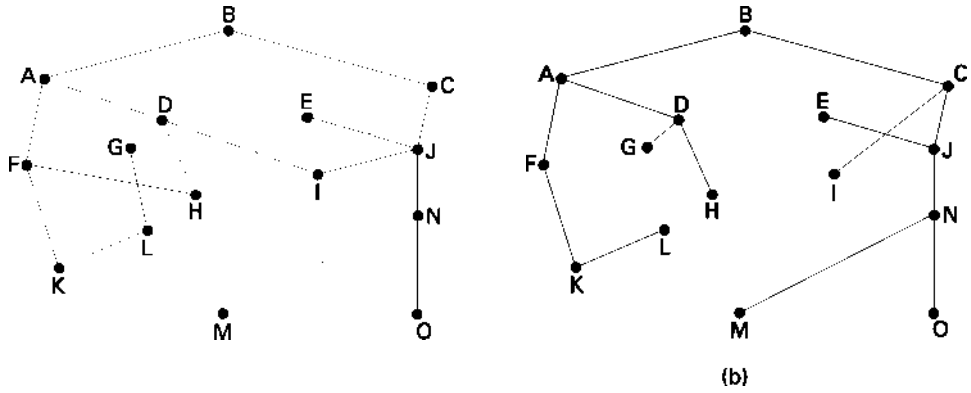


Fig. 5-6. (a) O subreţea. (b) Un arbore de scufundare pentru ruterul *B.*

Ca o consecinţă directă a principiului optimalităţii, putem observa că mulţimea rutelor de la toate sursele către o anume destinaţie formează un arbore având rădăcina în destinaţie. Acest arbore se numeşte **arbore de scufundare (sink tree)** şi este prezentat în fig. 5-6, unde distanţa metrica aleasă este numărul de salturi. Observaţi că arborele de scufundare nu este unic, putând exista şi alţi arbori cu aceeaşi lungime a căii. Scopul tuturor algoritmilor de dirijare este de a descoperi şi folosi arborii de scufundare pentru toate ruterele.

Deoarece arborele de scufundare este într-adevăr un arbore, el nu conţine bucle, deci fiecare pachet va fi livrat într-un număr finit şi limitat de salturi. În practică viaţa nu este chiar aşa de uşoară. Legăturile şi ruterele pot să se defecteze şi să-şi revină în timpul operaţiilor, astfel încât diferite rutere pot avea imagini diferite asupra topologiei curente. De asemenea, am trecut mai repede peste întrebarea

dacă fiecare ruter trebuie să obţină individual informaţia necesară calculării arborelui de scufundare sau dacă această informaţie este colectată prin alte mijloace. Vom reveni însă la această

problemă în curând. Cu toate acestea, principiul optimalităţii şi arborele de scufundare furnizează referinţe cu care pot fi comparaţi ceilalţi algoritmi de dirijare.

5.2.2 Dirijarea pe calea cea mai scurtă

Să începem studiul algoritmilor de dirijare posibili cu o tehnică des utilizată în multe forme deoarece este simplă şi uşor de înţeles. Ideea este de a construi un graf al subreţelei, fiecare nod al grafului fiind un ruter, iar fiecare arc al grafului fiind o linie de comunicaţie (numită adesea legătură). Pentru a alege o cale între o pereche dată de rutere, algoritmul trebuie să găsească în graf calea cea mai scurtă dintre ele.

Conceptul de **cea mai scurtă cale (shortest path routing)** necesită unele explicaţii. O modalitate de a măsura lungimea căii este numărul de salturi. Folosind această metrică, căile *ABC* şi *ABE* din fig. 5-7 sunt la fel de lungi. O altă metrică este distanţa geografică în kilometri, caz în care *ABC* este clar mult mai mare decât *ABE* (presupunând că fig. este desenată la scară).

Oricum, sunt posibile multe alte metrici în afară de salturi şi distanţa geografică. De exemplu, fiecare arc poate fi etichetat cu valorile medii ale aşteptării în coadă şi întârzierii de transmisie pentru anumite pachete standard de test, aşa cum sunt determinate de măsurători care se fac din oră în oră. Cu această etichetare, cea mai scurtă cale este cea mai rapidă, nu neapărat cea cu mai puţine arce sau kilometri.

B C (oo,

H (oo, -)

În cazul cel mai general, etichetele de pe arce ar putea fi calculate ca funcţii de distanţă, lărgime de bandă, trafic mediu, cost al comunicaţiei, lungime medie a cozilor de aşteptare, întârzieri măsurate şi alţi factori. Prin modificarea ponderilor, algoritmul ar putea calcula cea mai „scurtă” cale, în conformitate cu oricare dintre aceste criterii sau cu combinaţii ale acestor criterii.

Se cunosc mai mulţi algoritmi pentru calculul celei mai scurte căi între două noduri dintr-un graf. Cel mai cunoscut este cel propus de Dijkstra (1959). Fiecare nod este etichetat (în paranteze) cu distanţa de la nodul sursă până la el, de-a lungul celei mai bune căi cunoscute. Iniţial nu se cunoaşte nici o cale, aşa că toate nodurile vor fi etichetate cu infinit. Pe măsură ce se execută algoritmul şi se găsesc noi căi, etichetele se pot schimba, reflectând căi mai bune. O etichetă poate fi fie temporară, fie permanentă. Iniţial toate etichetele sunt temporare. Atunci când se descoperă că o etichetă reprezintă cea mai scurtă cale posibilă de la sursă către acel nod, ea devine permanentă şi nu se mai schimbă ulterior.

Pentru a ilustra cum funcţionează algoritmul de etichetare, să ne uităm la graful neorientat, etichetat din fig. 5-7(a), unde

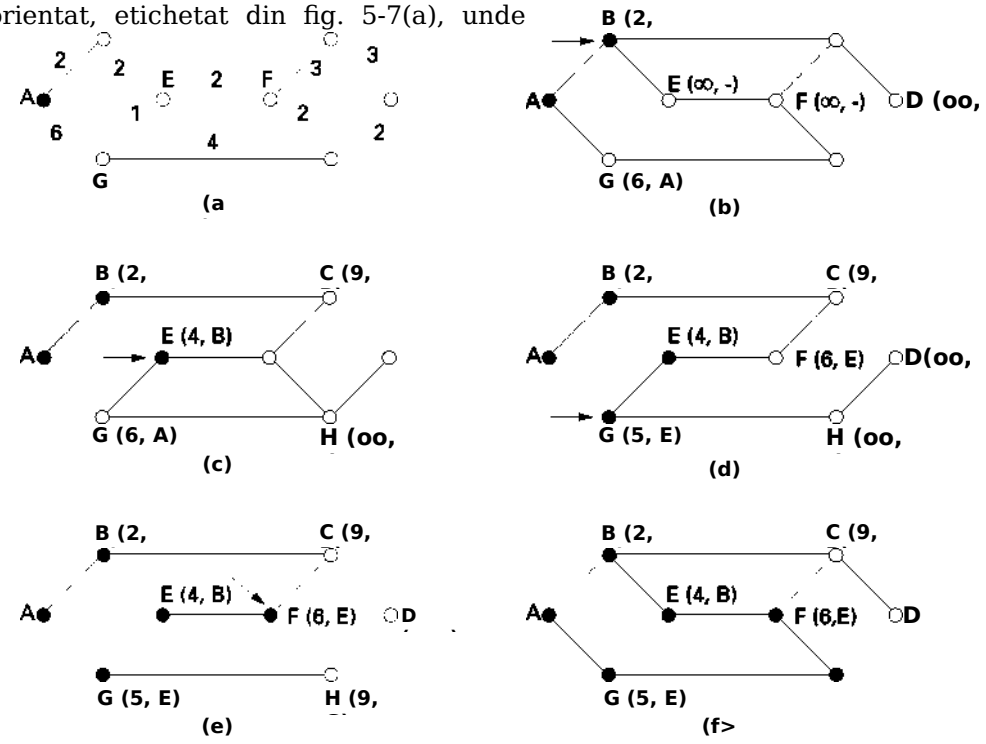


Fig. 5-7. Primii cinci paşi folosiţi în calcularea celei mai scurte căi de la A la D. Săgeţile indică nodul curent.

etichetele reprezintă, de exemplu, distanţa. Dorim să aflăm cea mai scurtă cale de

la *A* la *D.* Începem prin a marca nodul *A* ca permanent, indicând aceasta printr-un cerc colorat. Apoi vom examina fiecare nod adiacent cu *A* (care este acum nodul curent), reetichetând fiecare nod cu distanţa până la nodul *A*. De fiecare dată când un nod este reetichetat, îl vom eticheta şi cu nodul de la care s-a făcut încercarea, pentru a putea reface calea ulterior. După ce am examinat toate nodurile adiacente ale lui *A*, vom examina toate nodurile cu etichetă temporară din întregul graf şi îl facem permanent pe cel cu eticheta minimă, aşa cum se observă din fig. 5-7(b). Acest nod devine noul nod curent.

Acum începem din *B* şi examinăm toate nodurile sale adiacente. Dacă suma între eticheta lui *B* şi distanţa de la *B* la nodul considerat este mai mică decât eticheta acelui nod, înseamnă că am găsit o cale mai scurtă şi va trebui făcută reetichetarea nodului.

După ce toate nodurile adiacente nodului curent au fost inspectate şi au fost schimbate toate etichetele temporare posibile, se reia căutarea în întregul graf pentru a identifica nodul cu eticheta temporară minimă. Acest nod este făcut permanent şi devine nodul curent al etapei următoare. Fig. 5-7 prezintă primii cinci paşi ai algoritmului.

Pentru a vedea de ce merge algoritmul, să privim fig. 5-7(c). La momentul respectiv de abia am făcut permanent nodul *E.* Să presupunem că ar exista o cale mai scurtă decât *ABE,* de exemplu *AXYZE.* Există două posibilităţi: fie nodul *Z* a fost deja făcut permanent, fie încă nu a fost. Dacă a fost, atunci *E* a fost deja examinat (la pasul imediat următor celui la care *Z* a fost făcut permanent), astfel încât calea *AXYZE* nu a fost ignorată şi deci nu poate fi cea mai scurtă cale.

Să considerăm acum cazul în care *Z* este încă etichetă temporară. Atunci fie eticheta lui *Z* este mai mare sau egală cu cea a lui *E*, caz în care *ABE* nu poate fi o cale mai scurtă decât *AXYZE*, fie este mai mică decât cea a lui *E*, caz în care *Z* şi nu *E* va deveni permanent mai întâi, permiţând lui *E* să fie examinat din *Z*.

Algoritmul este prezentat în fig. 5-8. Variabilele globale *n* şi *dist* sunt iniţializate înainte să fie apelată *shortest_path.* Singura diferenţă între program şi algoritmul descris mai sus este aceea că în fig. 5-8 calculăm calea cea mai scurtă pornind de la nodul terminal, *t*, în locul nodului sursă, *s*. Deoarece calea cea mai scurtă de la *t* la *s* într-un graf neorientat este exact aceeaşi cu calea cea mai scurtă de la *s* la *t*, nu contează la care capăt începem (decât dacă există mai multe căi scurte, caz în care, inversând calea, am putea găsi alt drum). Motivul pentru care începem căutarea de la nodul destinaţie este acela că nodurile sunt etichetate cu predecesorul şi nu cu succesorul. Atunci când calea finală este copiată în variabila de ieşire, *path,* calea este inversată. Prin inversarea căutării, cele două efecte se anulează, astfel încât rezultatul se obţine în ordinea corectă.

```
#define MAX_NODES 1024                                  /* numărul maxim de noduri */
#define INFINITY 1000000000                          /* un număr mai mare decât orice cale
                                                     */ /* dist[i][j] e distanţa de la i la
[ MAX_NODES][ MAX_NODES]                             j */

void shortest_path(int s, int t, int path[])
{ struct state {                                            /* calea cu care se lucrează
*/
 int predecessor;                                                    /*  nodul  anterior
*/
 int length;                                         /* lungimea de la sursă la acest nod
*/
 enum {permanent, tentative} label;                                  /*  etichetă  stare
*/
}state [MAX_NODES];
int i, k, min;
struct state *p;

for (p = &state[0]; p < &state[n]; p++) {                            /* iniţializări */
 p->predecessor = -1;
 p->length = INFINITY;
 p->label = tentative;
}
state[t].length = 0; state[t].label = permanent;
k = t;                                                   /* k este nodul iniţial de lucru
*/
do {                                                    /* există vreo cale mai bună de la k?
*/
 for (i = 0; i < n; i++)                                            /* graful  are  n  noduri
*/
   if (dist[k][i] != 0 && state[i].label == tentative) {
     if (state[k].length + dist[k][i] < state[i].length) {
       state[i].predecessor = k;
       state[i] .length = state[k] .length + dist[k] [i];
     }
   }

 /* Găseşte nodul etichetat temporar cu cea mai mică etichetă */ k = 0; min = INFINITY;
 for (i = 0; i < n; i++)
   if (state[i].label == tentative && state[i].length < min) { min = state [i].length;
     k = i;
   }
 state[k].label = permanent;
} while (k != s);

/* Copiază calea în vectorul de ieşire */
i = 0; k = s;
do { path[i++] = k; k = state[k].predecessor; } while (k >= 0);
}
```

Fig. 5-8. Algoritmul Dijkstra pentru calculul celei mai scurte căi într-un graf.

5.2.3 Inundarea

Un alt algoritm static este **inundarea (flooding)**, în care fiecare pachet recepţionat este trimis mai departe pe fiecare linie de ieşire, cu excepţia celei pe care a sosit. Este evident că inundarea generează un mare număr de pachete duplicate, de fapt un număr infinit dacă nu se iau unele măsuri pentru a limita acest proces. O astfel de măsură este păstrarea unui contor de salturi în antetul fiecărui pachet, contor care este decrementat la fiecare salt şi care face ca pachetul să fie

distrus când contorul atinge valoarea zero.

Ideal ar fi ca acest contor să fie iniţializat cu lungimea căii de la sursă la destinaţie. Dacă emiţătorul nu cunoaşte lungimea căii, poate iniţializa contorul la valoarea cea mai defavorabilă, adică diametrul subreţelei.

O metodă alternativă pentru limitarea inundării este identificarea pachetelor care au fost deja inundate, pentru a preîntâmpina trimiterea lor a doua oară. O cale pentru a realiza acest scop este ca ruterul sursă să plaseze un număr de secvenţă în fiecare pachet pe care îl primeşte de la calculatorul gazdă asociat. Fiecare ruter necesită menţinerea unei liste pentru fiecare ruter sursă, cu numerele de secvenţă provenite de la acel ruter sursă şi care au fost deja trimise mai departe. Dacă soseşte un pachet care se află în listă, el nu mai este trimis mai departe.

Pentru a limita creşterea lungimii listei, fiecare listă trebuie însoţită de un contor, *k,* care semnifică faptul că toate numerele de secvenţă până la *k* au fost deja tratate. La recepţia unui pachet este uşor să se verifice dacă este un duplicat, caz în care este distrus. Evident, lista cu numere mai mici decât *k* nu este necesară, deoarece *k* o rezumă.

O variantă a algoritmului de inundare, care este şi ceva mai practică, este **inundarea selectivă** (**selective flooding**). În acest algoritm ruterele nu trimit fiecare pachet recepţionat pe fiecare legătură de ieşire, ci doar pe acele linii care duc aproximativ în direcţia potrivită. De obicei sunt puţine motive pentru a trimite un pachet spre partea de vest a reţelei folosind o legătură spre est, decât dacă topologia reţelei este cu totul deosebită şi ruterul este sigur de acest lucru.

Inundarea nu este practică pentru majoritatea aplicaţiilor, însă are destule utilizări. De exemplu, în aplicaţiile militare, unde un mare număr de rutere pot fi scoase din funcţionare în orice moment, robusteţea extraordinară a inundării este necesară. În aplicaţiile de baze de date distribuite, este uneori necesar ca toate bazele de date să fie actualizate simultan, caz în care inundarea poate fi folositoare. În reţelele fără fir, toate mesajele transmise de o gazdă pot fi recepţionate de toate celelalte gazde din raza sa radio, ceea ce înseamnă de fapt inundare, şi unii algoritmi folosesc această proprietate. O a patra utilizare posibilă a inundării este ca metrică la care să se raporteze toţi ceilalţi algoritmi de dirijare. Inundarea alege întotdeauna cea mai scurtă cale, deoarece alege în paralel toate căile posibile. În consecinţă, nici un alt algoritm nu poate produce o întârziere mai redusă (dacă ignorăm supraîncărcarea generată de însuşi procesul de inundare).

5.2.4 Dirijare cu vectori distanţă

Reţelele moderne de calculatoare folosesc de obicei algoritmi dinamici de dirijare în locul celor statici, descrişi anterior, deoarece algoritmii statici nu ţin seama de încărcarea curentă a reţelei. Doi dintre cei mai cunoscuţi algoritmi dinamici sunt algoritmul de dirijare cu vectori distanţă şi algoritmul de dirijare bazat pe starea legăturilor. În această secţiune ne vom ocupa de primul algoritm. În secţiunea următoare vom studia cel de-al doilea algoritm.

Algoritmul de **dirijare cu vectori distanţă (distance vector routing)** presupune că fiecare ruter menţine o tabelă (de exemplu un vector) care păstrează cea mai

bună distanţă cunoscută spre fiecare destinaţie şi linia care trebuie urmată pentru a ajunge acolo. Aceste tabele sunt actualizate prin schimbul de informaţii între nodurile vecine.

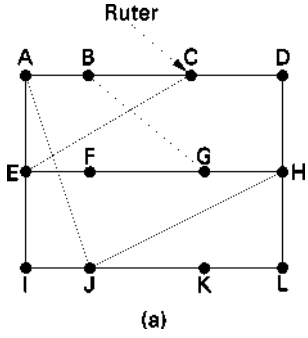
Algoritmul de dirijare cu vectori distanţă este cunoscut şi sub alte nume, cel mai des algoritmul distribuit de dirijare **Bellman-Ford** şi algoritmul **Ford-Fulkerson**, după numele cercetătorilor care l-

au propus (Bellman, 1957; şi Ford şi Fulkerson, 1962). A fost algoritmul de dirijare folosit iniţial în reţeaua ARPANET, a fost folosit de asemenea în Internet sub numele de RIP.

În dirijarea pe baza vectorilor distanţă, fiecare ruter păstrează o tabelă de dirijare conţinând câte o intrare pentru fiecare ruter din subreţea. Această intrare are două părţi: linia de ieşire preferată care se foloseşte pentru destinaţia respectivă şi o estimare a timpului sau distanţei până la acea destinaţie. Metrica folosită poate fi numărul de salturi, întârzierea în milisecunde, numărul total de pachete care aşteaptă în cozi de-a lungul căii, sau ceva asemănător.

Se presupune că ruterul cunoaşte „distanţa" spre fiecare dintre vecinii săi. Dacă se foloseşte metrica salturilor, distanţa este doar de un salt. Dacă metrica folosită este cea a lungimii cozilor de aşteptare, ruterul examinează pur şi simplu lungimile acestor cozi. Dacă metrica este cea a întârzierilor, ruterul o poate măsura direct prin pachete speciale ECHO, în care receptorul va marca doar timpul curent (ştampila de timp) şi le va trimite înapoi cât mai repede posibil.

Ca exemplu, să presupunem că se foloseşte metrica întârzierilor şi că ruterul cunoaşte întârzierea spre fiecare dintre vecinii săi. O dată la fiecare T msec fiecare ruter trimite spre fiecare vecin o listă a estimărilor proprii spre fiecare destinaţie. De asemenea el recepţionează o listă similară de la fiecare vecin. Să presupunem că una dintre aceste tabele tocmai a sosit de la vecinul X, cu X*{* fiind estimarea lui$X$ despre cât timp este necesar pentru a ajunge la ruterul i. Dacă ruterul ştie că întârzierea spre X este m msec, el ştie de asemenea că poate atinge ruterul i trecând prin X în Xj + m msec. Făcând aceste calcule pentru fiecare vecin, un ruter poate stabili care estimare pare a fi cea mai bună, pentru a folosi această estimare, împreună cu linia corespunzătoare în noua tabelă de dirijare. Este de remarcat faptul că vechea tabelă de dirijare nu intervine în calcule.



(a)

L	A	1	H	K	Noua întârziere	1 Linia
A	0	24	20	21	8	A
B	12	36	31	28	20	A
C	25	18	19	36	28	I
D	40	27	8	24	20	H
E	14	7	30	22	17	I
F	23	20	19	40	30	I
G	18	31	6	31	18	H
H	17	20	0	19	12	H
1	21	0	14	22	10	I
J	9	11	7	10	0	-
K	24	22	22	0	6	K
L	29	33	9	9	15	K

întîrzierile

JA	Jl	JH	JK
8	10	12	6

Noua tabelă de rutare

Vectorii primiţi de la

(b)

Fig. 5-9. (a) O subreţea. (b) Intrări de la *A, I, H* şi *K* şi noua tabelă de dirijare pentru *J.*

Acest proces de actualizare este ilustrat în fig. 5-9. Partea (a) prezintă o subreţea. Primele patru coloane din partea (b) conţin vectorii de întârzieri primiţi de la vecinii ruterului *J*. *A* afirmă că are 12 msec întârziere spre *B,* 25 msec întârziere spre *C,* 40 msec întârziere spre *D* etc. Presupunem că *J* şi-a măsurat sau estimat întârzierea faţă de vecinii săi *A, I, H* şi *K,* obţinând valorile 8, 10, 12 şi 16 ms, respectiv.

Să vedem cum calculează *J* noua cale spre ruterul *G.* El ştie că poate ajunge la *A* în 8 msec şi *A* pretinde că este în stare să ajungă la *G* în 18 msec, astfel încât *J* poate conta pe o întârziere de 26 msec spre G dacă dirijează pachetul spre *A*. Similar, el calculează întârzierea spre *G* prin *I, H, K* ca fiind 41 (31 + 10), 18 (6 + 12) şi 37 (31 + 6) respectiv. Cea mai bună valoare este 18, aşa că va crea o intrare în tabela de dirijare cu întârzierea către *G* de 18 msec şi ruta de urmat trecând prin *H*. Aceleaşi calcule se fac pentru toate destinaţiile, obţinându-se noua tabelă de dirijare, care este prezentată în ultima coloană a figurii.

Problema numărării la infinit

Dirijarea folosind vectori distanţă funcţionează în teorie, însă în practică are o limitare importantă: deşi ea converge spre rezultatul corect, o face foarte lent. În particular, ea reacţionează rapid la veştile bune, dar foarte lent la cele rele. S ă considerăm un ruter care are un cel mai bun drum spre destinaţia *X* foarte lung. Dacă la următorul schimb de informaţii, vecinul său *A* raportează brusc o întârziere mică spre *X*, ruterul va comuta şi va folosi linia spre *A* pentru a dirija traficul spre *X*. Astfel, într-o singură schimbare a vectorului, vestea bună a fost luată în considerare.

Pentru a vedea cât de repede se propagă veştile bune, să considerăm subreţeaua (liniară) de cinci noduri din fig. 5-10, unde metrica întârzierilor este numărul de salturi. Să presupunem că iniţial nodul *A* nu funcţionează şi toate celelalte rutere cunosc acest lucru. Cu alte cuvinte, toate celelalte rutere au înregistrat întârzierea spre *A* ca având valoarea infinit.

Când *A* porneşte, celelalte rutere află aceasta datorită schimbărilor din vector. Pentru simplificare, vom considera că există un gong uriaş undeva, care bate periodic pentru a iniţia schimbul de vectori simultan la toate ruterele. La momentul primului schimb, *B* află că vecinul din stânga are o întârziere nulă spre *A*. Astfel, *B* creează o nouă intrare în tabela sa, marcând faptul că *A* este la un singur salt distanţă, spre stânga. Toate celelalte rutere consideră că *A* este încă oprit. Intrările tabelei de dirijare pentru *A*, la acest moment, sunt prezentate în a doua linie din fig. 5-10(a). La următorul schimb, *C* află că *B* are o cale de lungime 1 spre *A* , astfel încât îşi actualizează tabela de dirijare pentru a indica o cale de lungime 2, însă *D* şi *E* nu vor primi vestea cea bună decât mai târziu. Evident, vestea cea bună se răspândeşte cu viteza de un salt la fiecare schimb. Într-o subreţea având calea cea mai lungă de lungime *N* salturi, după *N* schimburi fiecare ruter va afla despre liniile şi ruterele nou apărute.

A	B	C	D	E	
•	•	•	•	Iniţial
1	•	•	•	După 1 schimb
1	**2**	•	•	După 2 schimburi
1	**2**	3	•	După 3 schimburi
1	**2**	3	4	După 4 schimburi

(a)

A	B	C	D	E	
1		**2**	3	4	Iniţial
3		**2**	3	4	După 1 schimb
3		4	3	4	După 2 schimburi
5		4	5	4	După 3 schimburi
5		6	5	6	După 4 schimburi
7		6	7	6	După 5 schimburi
7		8	7	8	După 6 schimburi
•		•	•	•	

(b)

Fig. 5-10. Problema numărării la infinit.

Să considerăm acum situaţia din fig. 5-10(b), în care toate liniile şi ruterele sunt iniţial în funcţiune. Ruterele *B, C, D* şi *E* au distanţele spre *A* respectiv de 1, 2, 3, 4. Brusc, *A* se opreşte sau, alternativ, linia dintre *A* şi *B* este întreruptă, ceea ce reprezintă efectiv acelaşi lucru din punctul de vedere al lui *B*.

La primul schimb de pachete, *B* nu primeşte nimic de la *A*. Din fericire, *C* spune: „Nici o problemă. Eu ştiu o cale spre *A* de lungime 2.” Însă *B* nu ştie că această cale a lui *C* trece prin *B* însuşi. După cunoştinţele lui *B*, *C* ar putea avea zece linii, toate cu căi separate de lungime 2 spre *A*. Prin urmare *B* va crede că poate ajunge la *A* prin *C* pe o cale de lungime 3. *D* şi *E* nu îşi actualizează intrările proprii pentru *A* la primul schimb.

La al doilea schimb, *C* remarcă faptul că fiecare dintre vecinii săi pretinde a avea o cale de lungime 3 spre *A*. El va alege la întâmplare unul dintre aceşti vecini şi va înregistra noua distanţă spre *A* ca fiind 4, aşa cum se arată în linia a treia din fig. 5-10(b). Schimburile următoare vor produce succesiunea prezentată în continuare în fig. 5-10(b).

Din această figură se poate deduce de ce veştile rele circulă mai lent: orice ruter va avea întotdeauna o valoare cu cel mult unu mai mare decât valoarea minimă a vecinilor săi. Treptat, toate ruterele vor ajunge la infinit, însă numărul de schimburi necesar depinde de valoarea numerică folosită pentru a reprezenta valoarea infinit. Din această cauză este recomandat să se aleagă infinitul, ca fiind lungimea celei mai mari căi, plus 1. Dacă metrica este întârzierea în timp, atunci nu este definită nici o limită superioară, astfel încât este necesară o valoare mare pentru a preveni considerarea unui drum cu întârziere mare ca fiind un drum defect. Nu este deloc surprinzător că această problemă este numită **problema numărării la infinit** (**the count to infinity problem**). Au existat câteva încercări de rezolvare a problemei (cum ar fi orizont împărţit cu revers otrăvit – eng.: split horizon with poisoned reverse – în RFC 1058), dar nici una nu a funcţionat bine în general. Miezul problemei este că atunci când *X* îi spune lui *Y* că are o cale spre o destinaţie, *Y* nu are de unde să ştie dacă se află el însuşi pe acea cale.

5.2.5 Dirijarea folosind starea legăturilor

Dirijarea folosind vectori distanţă a fost folosită în ARPANET până în 1979, când a fost înlocuită prin dirijarea folosind starea legăturilor. Au fost două probleme importante care au cauzat această schimbare. În primul rând, deoarece metrica folosită era lungimea cozilor de aşteptare, la stabilirea rutei nu se lua în considerare lăţimea de bandă. Iniţial toate liniile erau de 56 Kbps, astfel încât lăţimea de bandă nu era o problemă, însă după ce câteva linii au fost îmbunătăţite la 230 Kbps, iar altele la 1.544 Mbps, neluarea în considerare a lăţimii de bandă a devenit o problemă majoră. Evident, era posibil să se schimbe metrica folosită pentru a depinde şi de lăţimea de bandă, însă exista şi o a doua problemă şi anume aceea că algoritmul convergea destul de greu (problema numărării la infinit). De aceea, a fost înlocuit cu un algoritm nou, numit algoritm de **dirijare folosind starea legăturilor (link state routing)**. Variante de dirijare folosind starea legăturilor sunt actualmente foarte răspândite.

Ideea algoritmului bazat pe starea legăturilor este simplă şi poate fi formulată în 5 puncte. Fiecare ruter trebuie să facă următoarele:

1. Să descopere care sunt vecinii săi şi afle adresele de reţea ale acestora.
2. Să măsoare întârzierea sau costul până la fiecare din vecinii săi.
3. Să pregătească un pachet prin care anunţă pe toată lumea că tocmai a terminat de cules datele despre vecini.

4. Să trimită acest pachet către toate celelalte rutere.
5. Să calculeze cea mai scurtă cale spre fiecare ruter.

Ca urmare, întreaga topologie şi toate întârzierile sunt măsurate experimental şi distribuite spre fiecare ruter. Apoi se poate rula algoritmul lui Dijkstra pentru a afla cea mai scurtă cale către fiecare ruter. În continuare vom analiza mai în detaliu aceşti cinci paşi.

Determinarea vecinilor

Când un ruter este pus în funcţiune, prima sa sarcină este să afle care sunt vecinii săi. El realizează aceasta prin trimiterea unui pachet special HELLO pe fiecare linie prin care este legat la alt ruter. Ruterul de la celălalt capăt trebuie să răspundă anunţând într-un pachet identitatea sa. Aceste nume trebuie să fie unice global, pentru că dacă mai târziu un ruter află că trei rutere sunt conectate toate la *F,* este esenţial ca acesta să poată determina dacă cele trei se referă la acelaşi *F.*

Când două sau mai multe rutere sunt conectate printr-o reţea LAN, situaţia devine puţin mai complicată. Fig. 5-11(a) ilustrează un LAN la care sunt conectate direct ruterele *A, C* şi *F*. Fiecare dintre aceste rutere este conectat cu unul sau mai multe alte rutere, aşa cum se arată în figură.

O modalitate de a modela reţeaua LAN este de a o considera ca un nod, aşa cum se arată în fig. 5-11 (b). Aici am introdus un nod nou, artificial, *N,* la care *A, C* şi *F* sunt conectate. Faptul că este posibil să se meargă de la *A* la *C* prin LAN este reprezentat aici de calea *ANC.*

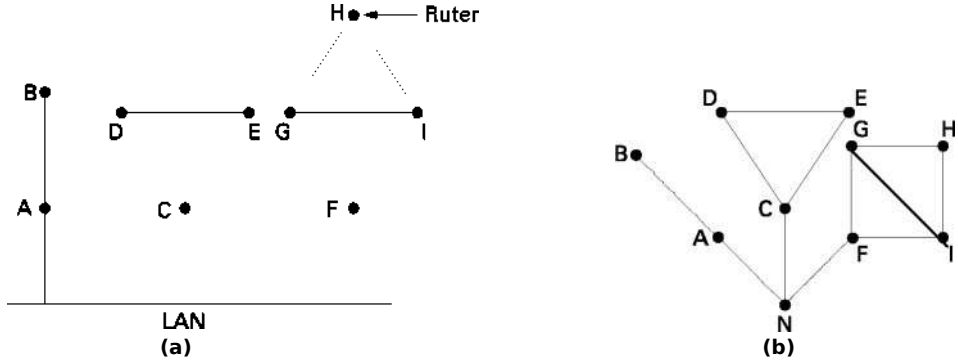


Fig. 5-11. (a) Nouă rutere şi o LAN. (b) Graful asociat punctului (a).

Măsurarea costului liniei

Algoritmul de dirijare bazat pe starea legăturilor cere ca fiecare ruter să ştie, sau cel puţin să aibă o estimare rezonabilă, a întârzierii către fiecare dintre vecinii săi. Cel mai direct mod de a afla acest lucru este de a trimite un pachet special ECHO pe linie, cerând ca ruterul partener să-l trimită înapoi imediat. Măsurând timpul în care pachetul se întoarce (round-trip time) şi împărţindu-l la doi, ruterul iniţiator poate avea o estimare rezonabilă a întârzierii. Pentru rezultate şi mai bune, testul poate fi repetat de mai multe ori, folosindu-se apoi valoarea medie obţinută. Bineînţeles, această metodă presupune implicit că întârzierile sunt simetrice, dar nu mereu este aşa.

O problemă interesantă este dacă să se considere sau nu încărcarea reţelei la

măsurarea întârzierii. Pentru a ţine cont de încărcare, timpul de revenire trebuie măsurat din momentul în care pachetul ECHO este pus în coadă. Pentru a ignora încărcarea, ceasul se poate porni în momentul în care pachetul ECHO ajunge pe prima poziţie din coadă.

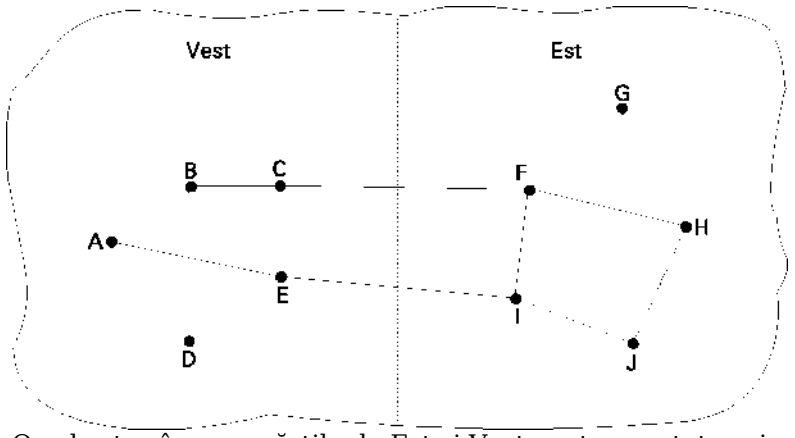


Fig. 5-12. O subreţea în care părţile de Est şi Vest sunt conectate prin două linii.

Pot fi aduse argumente în favoarea ambelor variante. Dacă se ţine cont de întârzierile provocate de trafic la măsurători înseamnă că dacă un ruter trebuie să aleagă între două linii cu aceeaşi lăţime de bandă, una dintre ele fiind puternic încărcată tot timpul, iar cealaltă nefiind foarte încărcată, atunci ruterul va desemna calea cea mai puţin încărcată ca fiind cea mai scurtă. Această alegere va duce la îmbunătăţirea performanţelor.

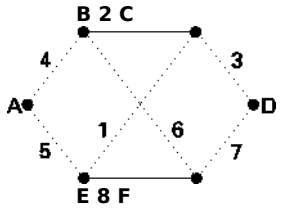
Din păcate, există şi un argument împotriva folosirii încărcării la calculul întârzierii. Să considerăm subreţeaua din fig. 5-12, care este divizată în două zone, Est şi Vest, conectate prin două linii, *CF* şi *EI.* Să presupunem că majoritatea traficului între Est şi Vest foloseşte linia *CF* şi, prin urmare, această linie este puternic încărcată şi are întârzieri mari. Folosirea întârzierilor în cozi pentru calculul celei mai scurte căi va face ca drumul *EI* să fie preferat. După ce noile tabele de dirijare au fost instalate, majoritatea traficului Est-Vest va trece acum prin *EI*, supraîncărcând-o. De aceea, la următoarea actualizare, *CF* va părea a fi calea cea mai scurtă. Prin urmare tabelele de dirijare vor oscila puternic, conducând la o dirijare fluctuantă şi facilitând apariţia multor probleme potenţiale. Dacă nu se ţine cont de încărcare, luându-se în considerare doar lăţimea de bandă, această problemă nu mai apare. Alternativ, încărcarea poate fi distribuită pe ambele linii, dar această soluţie nu foloseşte în întregime calea cea mai bună. Cu toate acestea, pentru a evita oscilaţiile în alegerea celei mai bune căi, poate fi înţelept să se distribuie încărcarea pe mai multe linii, în proporţii cunoscute pe fiecare linie.

Construirea pachetelor cu starea legăturilor

De îndată ce a fost colectată informaţia necesară pentru realizarea schimbului, se poate trece la pasul următor, fiecare ruter construind un pachet care conţine toate datele. Pachetul începe cu identitatea expeditorului, urmată de un număr de

secvenţă, vârstă (care va fi descrisă în continuare) şi o listă a vecinilor. Pentru fiecare vecin se specifică întârzierea asociată. Un exemplu de subreţea este prezentat în fig. 5-13(a), unde etichetele liniilor reprezintă întârzierile asociate. Pachetele cu starea legăturilor asociate tuturor celor şase rutere sunt prezentate în fig. 5-13(b).

Construirea pachetelor cu starea legăturilor se face uşor. Partea mai dificilă este să se determine când să fie construite ele. O posibilitate este ca ele să fie construite periodic, adică la intervale regulate. O altă posibilitate este atunci când se produce un eveniment semnificativ, cum ar fi scoaterea din funcţiune a unui vecin sau a unei linii, sau repunerea lor în funcţiune, sau modificarea semnificativă a proprietăţilor lor.



(a)

Pachete cu starea

A	
Secv.	
Vârst	
B	4
E	5

B	
Secv.	
Vârst	
A	4
C	2
F	6

C	
Secv.	
Vârst	
B	2
D	3
E	1

D	
Secv.	
Vârst	
C	3
F	7

E	
Secv.	
Vârst	
A	5
C	1
F	8

F	
Secv.	
Vârst	
B	6
D	7
E	8

(b

Fig. 5-13. (a) O subreţea. (b) Pachetele cu starea legăturilor pentru această subreţea.

Distribuirea pachetelor cu starea legăturilor

Cea mai complicată parte a algoritmului este distribuirea sigură a pachetelor cu starea legăturilor. De îndată ce pachetele sunt distribuite şi instalate, ruterele care primesc primele pachete îşi vor schimba rutele. În consecinţă, rutere diferite ar putea folosi versiuni diferite ale topologiei, ceea ce poate duce la apariţia unor inconsistenţe, bucle, maşini inaccesibile sau a altor probleme.

Pentru început vom descrie algoritmul de bază folosit pentru distribuţie. Apoi vom prezenta unele îmbunătăţiri posibile. Ideea fundamentală este folosirea inundării pentru a distribui pachetele cu starea legăturilor. Pentru a avea controlul inundării, fiecare pachet conţine un număr de secvenţă care este incrementat la fiecare nou pachet trimis. Ruterele păstrează evidenţa tuturor perechilor (ruter sursă, număr secvenţă) pe care le văd. La sosirea unui nou pachet cu starea legăturilor, el este căutat în lista pachetelor deja văzute. Dacă pachetul este nou, el este retrimis pe toate liniile, cu excepţia celei pe care a sosit. Dacă este un duplicat, pachetul este distrus. Dacă pachetul sosit are un număr de secvenţă mai mic decât cel mai mare număr de secvenţă detectat, atunci el este rejectat ca fiind învechit.

Acest algoritm are câteva probleme, dar ele sunt tratabile. În primul rând, dacă numerele de secvenţă ating valoarea maximă posibilă şi reîncep de la valori mici, se pot produce confuzii. Soluţia este folosirea numerelor de secvenţă pe 32 biţi. Considerând un pachet de starea legăturilor pe secundă, ar trebui 137 ani pentru a se reîncepe de la valori mici, astfel încât această posibilitate poate fi ignorată.

În al doilea rând, dacă un ruter se defectează, el va pierde evidenţa numerelor de secvenţă. Dacă va reîncepe de la 0, următorul pachet va fi rejectat ca duplicat.

În al treilea rând, dacă numărul de secvenţă este alterat şi se recepţionează 65540 în loc de 4 (eroare de modificare a unui bit), pachetele de la 5 la 65540 vor fi rejectate ca fiind învechite, deoarece se consideră că numărul de secvenţă curent este 65540.

Soluţia tuturor acestor probleme este includerea vârstei pachetului după numărul de secvenţă şi decrementarea sa la fiecare secundă. Când vârsta ajunge la zero, informaţia de la ruterul respectiv este distrusă. În mod normal un nou pachet apare, să zicem, la fiecare 10 sec., astfel încât informaţia de la ruter expiră doar dacă ruterul este oprit (sau dacă şase pachete consecutive s-au pierdut, un lucru extrem de improbabil). Câmpul *Age* este decrementat de asemenea de fiecare ruter la începutul procesului de inundare, pentru a se asigura că nici un pachet nu se poate pierde sau nu poate supravieţui o perioad ă nelimitată (un pachet având

vârsta zero este distrus).

Unele îmbunătăţiri ale algoritmului pot să-l facă mai robust. Atunci când un pachet cu starea legăturilor ajunge într-un ruter pentru inundare, acesta nu este pus imediat în coada de transmisie. El este pus într-o zonă de aşteptare pentru o scurtă perioadă. Dacă înainte de a fi trimis primul pachet soseşte un alt pachet cu starea legăturilor de la aceeaşi sursă, numerele lor de secvenţă sunt comparate. Dacă sunt identice, duplicatul este distrus. Dacă sunt diferite, cel mai bătrân este ignorat. Pentru a preîntâmpina erorile pe linia ruter-ruter, toate aceste pachete sunt confirmate. Dacă linia nu este folosită, zona de aşteptare este inspectată în manieră round-robin, pentru a selecta un pachet sau o confirmare de trimis.

Structura de date folosită de ruterul *B* pentru subreţeaua din fig. 5-13(a) este descrisă în fig. 5-46. Fiecare linie corespunde unui pachet cu starea legăturilor sosit recent, dar încă neprelucrat în întregime. În tabelă se înregistrează originea pachetului, numărul de secvenţă, vârsta şi datele transportate. În plus mai există unele indicatoare (flags) de trimitere şi confirmare pentru fiecare dintre cele trei linii ale lui *B* (respectiv către *A, C, F).* Indicatorul de trimitere precizează că pachetul trebuie trimis pe linia indicată. Indicatorul de confirmare precizează că respectivul pachet trebuie confirmat.

În fig. 5-14, pachetul cu starea legăturilor de la *A* soseşte direct, astfel încât el trebuie trimis către *C* şi *F* şi trebuie confirmat către *A*, aşa cum precizează biţii indicatori. Similar, pachetul de la *F* trebuie trimis către *A* şi *C* şi confirmat către *F*.

A	**21**	**60**	**0**	**1**	**1**	**1**	**0**	**0**	
F	**21**	**60**	**1**	**1**	**0**	**0**	**0**	**1**	
E	**21**	**59**	**0**	**1**	**0**	**1**	**0**	**1**	
C	**20**	**60**	**1**	**0**	**1**	**0**	**1**	**0**	
D	**21**	**59**	**1**	**0**	**0**	**0**	**1**	**1**	
				Trimitere			**ACK**		
Sursa	**Secv.**	**Vârsta**	**A**	**C**	**F**		**A**	**C**	**F**	**Date**

Oricum, situaţia celui de-al treilea pachet, de la *E,* este diferită. El soseşte de două ori, întâi pe calea *EAB* şi apoi pe calea *EFB.* În consecinţă el trebuie trimis numai către *C,* însă trebuie confirmat atât lui *A* cât şi lui *F,* aşa cum indică şi biţii corespunzători.

Dacă soseşte un duplicat în timp ce pachetul este încă în zona tampon, biţii trebuie modificaţi. De exemplu, dacă o copie a stării lui *C* soseşte de la *F* înainte ca a patra intrare din tabelă să fie trimisă, cei şase biţi se vor schimba în 100011,

Fig. 5-14. Zona tampon pentru pachete a ruterului *B* din fig. 5-13.

pentru a preciza că pachetul trebuie confirmat către *F*, dar nu trimis acolo.

Calcularea noilor rute

De îndată ce un ruter a acumulat un set complet de pachete cu starea

legăturilor, el poate construi graful întregii subreţele, deoarece fiecare legătură este reprezentată. De fapt, fiecare legătură este reprezentată de două ori, o dată pentru fiecare direcţie. Cele două valori pot fi mediate sau folosite separat.

Acum poate fi folosit local algoritmul lui Dijkstra, pentru a construi cea mai scurtă cale către toate destinaţiile posibile. Rezultatul acestui algoritm poate fi trecut în tabelele de dirijare, iar apoi operaţiile normale pot fi reluate.

Pentru o subreţea formată din n rutere, fiecare având k vecini, memoria necesară pentru a memora datele de intrare este proporţională cu kn. Pentru subreţele mari, aceasta poate fi o problemă. De asemenea şi timpul de calcul poate fi important. Cu toate acestea, în multe situaţii, algoritmul bazat pe starea legăturilor funcţionează bine.

Oricum, probleme ale hardware-ului sau software-ului pot influenţa negativ funcţionarea acestui algoritm (la fel ca şi a altora). De exemplu, dacă un ruter pretinde că are o linie pe care de fapt nu o are, sau uită despre o linie pe care o are, graful subreţelei va fi incorect. Dacă un ruter eşuează în retrimiterea pachetelor sau le alterează în timp ce le retrimite, vor apărea probleme. În fine, dacă un ruter rămâne fără memorie sau calculează greşit rutele, se vor petrece lucruri urâte. Pe măsură ce subreţeaua creşte în dimensiune, ajungând la zeci sau sute de mii de noduri, probabilitatea ca un ruter să se defecteze devine neneglijabilă. Trucul care se poate folosi constă în încercarea de limitare a pagubelor atunci când inevitabilul s-a produs. Perlman (1988) tratează în detaliu aceste probleme, precum şi soluţiile lor.

Dirijarea bazată pe starea legăturilor este larg folosită în reţelele actuale, astfel încât ar fi potrivite câteva cuvinte despre unele protocoale care o folosesc. Protocolul OSPF, care este extrem de folosit în Internet, utilizează un algoritm bazat pe starea legăturilor. Vom descrie protocolul OSPF în sect. 5.6.4.

Un alt protocol bazat pe starea legăturilor este **IS-IS** (**Intermediate System-Intermediate System,** Sistem Intermediar – Sistem Intermediar), care a fost proiectat pentru DECnet şi mai apoi adoptat de ISO pentru a fi folosit cu protocolul neorientat pe conexiune de la nivelul reţea, CLNP. De atunci a fost modificat pentru a se descurca şi cu alte protocoale, cel mai important fiind IP. ISIS este folosit în coloane vertebrale ale Internet-ului (Internet backbones) (inclusiv în vechiul NSFNET) şi în unele sisteme digitale celulare cum ar fi CDPD. Novell NetWare foloseşte o variantă simplificată IS-IS (NLSP) pentru a dirija pachetele IPX.

În principiu IS-IS distribuie o imagine a topologiei ruterelor, pe baza căreia se calculează calea cea mai scurtă. Fiecare ruter anunţă, în informaţia de stare a legăturilor sale, ce adrese la nivelul reţea poate să acceseze direct. Aceste adrese pot fi IP, IPX, AppleTalk, sau orice alte adrese. IS-IS poate accepta chiar mai multe protocoale ale nivelului reţea în acelaşi timp.

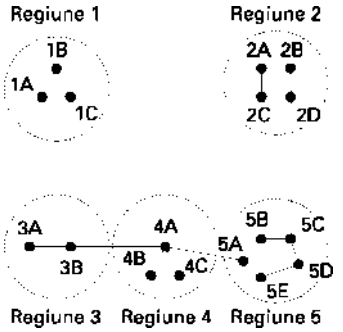
Multe din inovaţiile din proiectarea lui IS-IS au fost preluate de OSPF (OSPF a fost proiectat la mulţi ani după IS-IS). Acestea includ o metodă auto-stabilizatoare a inundării, folosită la actualizarea stării legăturilor, conceptul de ruter dedicat într-o LAN, metoda de calcul şi utilizare a căilor divizate şi mai multe metrici. Ca o consecinţă, vor exista puţine diferenţe între IS-IS şi OSPF. Cea mai importantă diferenţă este că IS-IS este codificat în aşa fel, încât suportă uşor şi chiar natural să

transporte informaţiile mai multor protocoale ale nivelului reţea, o caracteristică pe care OSPF nu o are. Acest avantaj este deosebit de important, în special în mediile mari, multiprotocol.

5.2.6 Dirijare ierarhică

Pe măsură ce reţelele cresc în dimensiune, tabelele de dirijare cresc proporţional. Pe lângă faptul că memoria ruterului este consumată de fiecare nouă creştere a tabelelor, pentru parcurgerea lor este necesar tot mai mult timp de calcul şi se foloseşte tot mai mult din lăţimea de bandă pentru a trimite rapoartele de stare despre ele. La un moment dat, reţeaua poate creşte până la un punct în care nu mai este posibil ca fiecare ruter să deţină o intrare pentru fiecare alt ruter, astfel încât dirijarea trebuie făcută ierarhic, la fel ca în reţeaua telefonică.

Atunci când se foloseşte dirijarea ierarhică, ruterele sunt împărţite în ceea ce vom numi **regiuni** (**regions**), fiecare ruter ştiind toate detaliile necesare pentru a dirija pachete spre destinaţie în cadrul regiunii sale, dar neştiind nimic despre organizarea internă a celorlalte regiuni. Când reţele diferite sunt interconectate, este natural să privim fiecare reţea ca pe o regiune, pentru a elibera ruterele dintr-o reţea de sarcina de a cunoaşte structura topologică a celorlalte.



(a

Tabela completă pentru 1A Deşt. Linie

Saltu	—	—
IA	1B	1
1B	1C	1
IC	1B	2
2A	1B	3
2B	1B	3
2C	1B	4
2D	1C	3
3A	1C	2
3B	1C	3
4A	1C	4
4B	1C	4
4C	1C	4
5A	1C	5
5B	1B	5
5C	1C	6
5D	1C	5

(b

Tabela ierarhică pentru 1A Deşt. Linie Salturi

—	—
1B	1
1C	1
1B	2
1C	2
1C	3
1C	4

Ic)

Fig. 5-15. Dirijare ierarhică.

Pentru reţele uriaşe, o ierarhie pe două niveluri ar putea fi insuficientă, ar putea fi necesar să grupăm regiunile în asociaţii (clusters), asociaţiile în zone, zonele în grupuri şi aşa mai departe până ce nu mai avem nume pentru aceste aglomerări. Ca un exemplu de ierarhie multinivel, să considerăm cum poate fi dirijat un pachet din Berkeley, California spre Malindi, Kenya. Ruterul din Berkeley cunoaşte în detaliu topologia din California, aşa că va trimite tot traficul pentru exteriorul statului către ruterul din Los Angeles. Ruterul din Los Angeles este capabil să dirijeze traficul spre alte rutere interne, dar va trimite tot traficul extern spre New York. Ruterul din New York este programat să dirijeze traficul către ruterul din ţara de destinaţie care se ocupă de traficul extern, de exemplu în Nairobi. În final, pachetul va urma calea sa, coborând în arborele din Kenya până ce ajunge la Malindi.

Fig. 5-15 sugerează un exemplu cantitativ al dirijării într-o ierarhie pe două niveluri, având cinci regiuni. Tabela de dirijare completă a ruterului *1A* are 17 intrări, aşa cum se arată în fig. 5-15 (b). Atunci când dirijarea se face ierarhic, aşa ca în fig. 5-15(c), există intrări pentru toate ruterele locale, la fel ca şi până acum, însă toate celelalte regiuni au fost condensate într-un singur ruter, astfel încât tot traficul pentru regiunea 2 va merge pe linia *1B-2A,* iar restul traficului la distanţă va merge pe linia 1*C*-3*B*. Dirijarea ierarhică a redus dimensiunea tabelei de la 17 intrări la 7. Pe măsură ce raportul între numărul de regiuni şi numărul de rutere dintr-o regiune creşte, se economiseşte tot mai mult spaţiu de memorie.

Din păcate acest câştig de spaţiu nu este gratuit. Trebuie plătit un preţ şi acesta este concretizat în creşterea lungimii căilor. De exemplu, cea mai bună cale de la *1A* la 5*C* este prin regiunea 2, însă în dirijarea ierarhică tot traficul către regiunea 5 este trimis spre regiunea 3, deoarece aşa este mai bine pentru majoritatea destinaţiilor din regiunea 5.

Dacă o reţea unică devine prea mare, o întrebare interesantă este: Câte niveluri trebuie să aibă ierarhia? De exemplu, să considerăm o subreţea cu 720 rutere. În absenţa oricărei ierarhii, tabela de dirijare a fiecărui ruter trebuie să conţină 720 intrări. Dacă subreţeaua este partiţionată în 24 regiuni cu 30 de rutere fiecare, fiecare ruter necesită 30 de intrări locale plus 23 de intrări pentru celelalte regiuni, deci un total de 53 intrări. Dacă se alege o ierarhie pe trei niveluri, cu opt asociaţii (clusters), fiecare având 9 regiuni a câte 10 rutere fiecare, fiecare ruter are nevoie de 10 intrări pentru ruterele locale, 8 intrări pentru celelalte regiuni din asociaţia sa şi de 7 intrări pentru celelalte asociaţii, deci un total de 25 intrări. Kamoun şi Kleinrock (1979) au descoperit că numărul optim de niveluri pentru o subreţea cu N rutere este $\ln N$, ceea ce necesită un total de $e \ln N$ intrări pentru fiecare ruter. De asemenea ei au arătat că creşterea efectivă a lungimii medii a căilor provocată de dirijarea ierarhică este suficient de mică pentru a fi acceptată.

5.2.7 Dirijarea prin difuzare

În unele aplicaţii, calculatoarele gazdă au nevoie să trimită mesaje către mai multe sau către toate celelalte calculatoare gazdă. De exemplu, un serviciu de distribuire a rapoartelor meteorologice, de actualizare a cursului acţiunilor sau transmisiuni radio în direct ar putea funcţiona mai bine prin difuzarea datelor către toate maşinile, fiind la latitudinea celor interesate de date să le recepţioneze. Trimiterea simultană a unui pachet către toate destinaţiile se numeşte **difuzare** (**broadcast**); mai multe metode pentru realizarea ei au fost propuse.

O metodă de difuzare care nu are cerinţe speciale pentru subreţea este ca sursa să trimită un pachet distinct către fiecare destinaţie. Metoda este nu numai consumatoare de lăţime de bandă, dar ea cere ca sursa să deţină o listă completă a tuturor destinaţiilor. S-ar putea ca în practică aceasta să fie singura metodă utilizabilă, însă ea este metoda cea mai puţin dorită.

Inundarea este un alt candidat evident. Deşi inundarea este nepotrivită pentru comunicaţia obişnuită capăt la capăt, pentru difuzare ar trebui luată serios în considerare, mai ales dacă nici una dintre metodele ce vor fi descrise în continuare nu este aplicabilă. Problema folosirii inundării ca metodă de difuzare este aceeaşi cu problema ivită la folosirea sa ca algoritm de dirijare capăt la capăt: generează prea multe pachete şi consumă lăţime de bandă prea mare.

Al treilea algoritm este **dirijarea multidestinaţie** (**mult.idest.inat.ion routing**). Dacă se foloseşte această metodă, fiecare pachet conţine fie o listă a destinaţiilor, fie o hartă de biţi care indică destinaţiile dorite. Atunci când un pachet ajunge la un ruter, ruterul verifică toate destinaţiile pentru a determina setul liniilor de ieşire pe care trebuie trimis pachetul. (O linie de ieşire este selectată dacă este calea cea mai bună pentru cel puţin o destinaţie.) Ruterul generează o nouă copie a pachetului pentru fiecare linie de ieşire folosită şi include în fiecare pachet doar acele destinaţii care folosesc linia respectivă. Efectul este partiţionarea mulţimii destinaţiilor între liniile de ieşire. După un număr suficient de salturi, fiecare pachet va conţine o singură destinaţie şi poate fi tratat ca un pachet normal. Dirijarea multidestinaţie este asemănătoare trimiterii separate a mai multor pachete către

adresele destinaţie, cu excepţia faptului că atunci când mai multe pachete trebuie să urmeze aceeaşi cale, unul dintre ele plăteşte tot drumul, iar celelalte călătoresc gratuit.

Al patrulea algoritm de difuzare utilizează explicit arborele de scufundare al ruterului care iniţiază difuzarea sau orice alt arbore de acoperire util. Un **arbore de acoperire** (**spanning tree**) este un subset al subreţelei care include toate ruterele şi nu conţine bucle. Dacă fiecare ruter cunoaşte care din liniile sale participă la arborele de acoperire, el poate copia un pachet de difuzare recepţionat pe toate liniile de ieşire care fac parte din arborele de acoperire, cu excepţia celei pe care a fost recepţionat. Această metodă asigură o utilizare deosebit de eficientă a lăţimii de bandă, generând numărul minim de pachete necesare pentru a rezolva problema. Singura problemă este că, pentru a fi aplicabilă, fiecare ruter trebuie să deţină cunoştinţe despre un anume arbore de acoperire. Uneori această informaţie este disponibilă (de exemplu la dirijarea bazată pe starea legăturilor), iar alteori nu este disponibilă (de exemplu la dirijarea cu vectori distanţă).

Ultimul nostru algoritm cu difuzare este o încercare de a aproxima comportamentul precedentului, chiar şi atunci când ruterele nu ştiu absolut nimic despre arborele de acoperire. Ideea, numită **trimiterea pe calea inversă** (**reverse path forwarding**) este remarcabil de simplă odată ce a fost indicată. Când un pachet de difuzare ajunge la un ruter, acesta verifică dacă pachetul a sosit pe linia pe care se trimit de obicei pachete către sursa difuzării. Dacă este aşa, este o şansă foarte mare ca însuşi pachetul de difuzare să fi urmat cea mai bună cale, fiind astfel prima copie care ajunge la ruter. Aceasta fiind situaţia, ruterul trimite pachetul pe toate liniile de ieşire, cu excepţia celei pe care a sosit. Dacă însă pachetul a sosit pe altă linie decât cea preferată pentru a ajunge la sursă, pachetul este distrus, fiind considerat un posibil duplicat.

Un exemplu de **trimitere pe calea inversă**, este prezentat în fig. 5-16. Partea (a) prezintă subre- ţeaua, partea (b) arată arborele de scufundare pentru ruterul *I* al subreţelei, iar partea (c) prezintă cum funcţionează algoritmul căii inverse. La primul salt, *I* trimite pachete către *F, H, J* şi *N,* aşa cum se vede din al doilea nivel al arborelui. Fiecare din aceste pachete ajunge pe calea preferată către *I* (presupunând că ruta preferată face parte din arborele de scufundare), lucru care este indicat printrun cerc în jurul literei. La saltul următor, se generează opt pachete, două de către fiecare ruter care a primit un pachet la primul salt. Aşa cum se vede, toate aceste opt pachete ajung la rutere încă nevizitate şi cinci dintre ele ajung pe linia preferată. Dintre cele şase pachete generate la al treilea salt, doar trei sosesc pe linia preferată (în *C, E* şi *K);* celelalte sunt duplicate. După cinci salturi şi 24 pachete, difuzarea se termină, spre deosebire de patru salturi şi 14 pachete necesare dacă arborele de scufundare ar fi fost urmat integral.

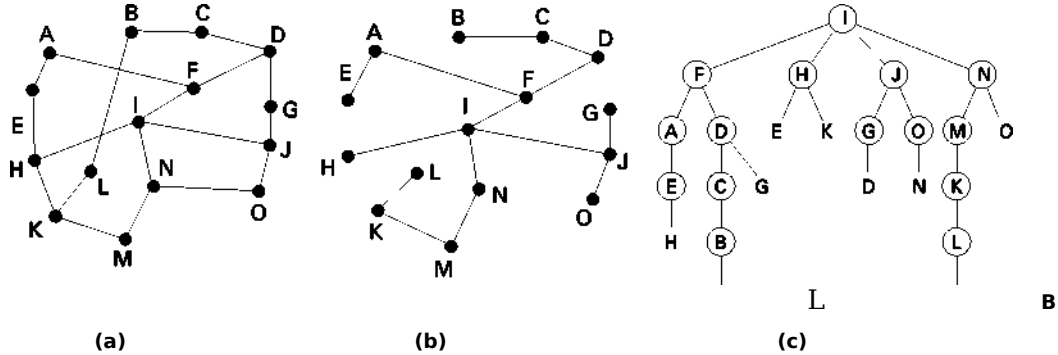


Fig. 5-16. Algoritmul trimiterii pe calea inversă. (a) O subreţea. (b) Un arbore de scufundare.
(c) Arborele construit de algoritmul căii inverse.

Principalul avantaj al folosirii căii inverse este acela că este rezonabil de eficient şi este uşor de implementat. El nu cere ruterelor să cunoască arborele de acoperire şi nici nu are overhead-ul algoritmului de adresare multidestinaţie, care foloseşte o listă de destinaţii sau hartă de biţi la fiecare difuzare. De asemenea, el nu necesită nici un mecanism special pentru a opri procesul, ca în cazul inundării (unde se foloseşte fie un contor al salturilor în fiecare pachet şi o cunoaştere a priori a diametrului subreţelei, fie o listă de pachete deja văzute pentru fiecare sursă).

5.2.8 Dirijarea cu trimitere multiplă (multicast)

Unele aplicaţii necesită ca procese aflate la mari distanţe unele de altele să lucreze în grup, de exemplu, un grup de procese care implementează o bază de date distribuită. În aceste situaţii, este adesea necesar ca un proces să trimită un mesaj către toţi ceilalţi membri ai grupului. Dacă grupul este mic, el poate trimite fiecărui partener un mesaj capăt la capăt. Dacă grupul este mare, această strategie este costisitoare. Uneori se poate folosi difuzarea, dar folosirea difuzării pentru a anunţa 1000 de maşini dintr-o reţea cu un milion de noduri este ineficientă, deoarece majoritatea receptorilor nu sunt interesaţi de mesaj (sau chiar mai rău, sunt foarte interesaţi, dar nu trebuie să vadă mesajul). De aceea, avem nevoie de o modalitate de a trimite mesaje spre grupuri bine definite, care conţin un număr mare de noduri, dar totuşi redus faţă de dimensiunea întregii reţele.

Trimiterea unui mesaj către un astfel de grup se numeşte **multicasting**, iar algoritmul de dirijare asociat se numeşte **dirijare multicast** (**multicast routing**). În această secţiune, vom descrie o modalitate de a realiza dirijarea multicast. Pentru informaţii suplimentare, vezi (Chu et al., 2000; Costa et al. 2001; Kasera et al., 2000; Madruga and Garcia-Luna-Aceves, 2001; Zhang and Ryu, 2001).

Dirijarea multicast necesită managementul grupului. Trebuie să existe modalităţi de a crea şi distruge grupuri şi de a permite proceselor să intre în grupuri sau să le părăsească. Cum se realizează aceste funcţii nu este treaba algoritmului de dirijare. Important pentru algoritmul de dirijare este că atunci când un proces se ataşează unui grup, el trebuie să informeze gazda sa despre aceasta. Este important ca ruterele să ştie căror grupuri le aparţin calculatoarele gazdă asociate. Fie

calculatoarele gazdă trebuie să anunţe ruterul asociat la producerea unei modificări în alcătuirea grupurilor, fie ruterul trebuie să interogheze periodic aceste calculatoare gazdă. În ambele cazuri, ruterul află din ce grupuri fac parte calculatoarele gazdă. Ruterele îşi informează vecinii, astfel că informaţia se propagă prin subreţea.

Pentru a realiza dirijarea multicast, fiecare ruter calculează arborele de acoperire care acoperă toate celelalte rutere din subreţea. De exemplu, în fig. 5-17(a) avem două grupuri, 1 şi 2. Unele rutere sunt ataşate la calculatoare gazdă care aparţin unuia sau ambelor grupuri, aşa cum se indică în figură. Un arbore de acoperire pentru cel mai din stânga ruter este prezentat în fig. 5-17(b).

Atunci când un proces trimite un pachet multicast către un grup, primul ruter îşi examinează arborele de acoperire şi îl retează, eliminând toate liniile care nu conduc către calculatoare gazdă, membre ale grupului. În exemplul nostru, fig. 5-17(c) arată arborele de acoperire retezat pentru grupul 1. Similar, fig. 5-17(d) arată arborele de acoperire retezat pentru grupul 2. Pachetele multicast sunt dirijate doar de-a lungul arborelui de acoperire corespunzător.

Sunt posibile mai multe moduri de retezare a arborelui de acoperire. Cel mai simplu se poate folosi dacă se utilizează dirijarea bazată pe starea legăturilor şi fiecare ruter cunoaşte întreaga topologie a subreţelei, inclusiv apartenenţa calculatoarelor gazdă la grupuri. Atunci arborele poate fi retezat pornind de la sfârşitul fiecărei căi, mergând spre rădăcină şi eliminând toate ruterele care nu aparţin grupului respectiv.

În cazul dirijării folosind vectori distanţă, poate fi aplicată o altă strategie de retezare a arborelui. Algoritmul de bază folosit este trimiterea pe calea inversă. Oricum, ori de câte ori un ruter fără nici un calculator gazd ă interesat de un anume grup şi fără nici o conexiune la alte rutere primeşte un mesaj multicast pentru acel grup, el va răspunde cu un mesaj PRUNE (retezare), anunţând expeditorul să nu îi mai trimită mesaje multicast pentru acel grup. Când un ruter care nu are printre calculatoarele gazdă nici un membru al vreunui grup primeşte astfel de mesaje pe toate liniile sale, el poate de asemenea să răspundă cu un mesaj PRUNE. În acest fel subreţeaua este retezată recursiv.

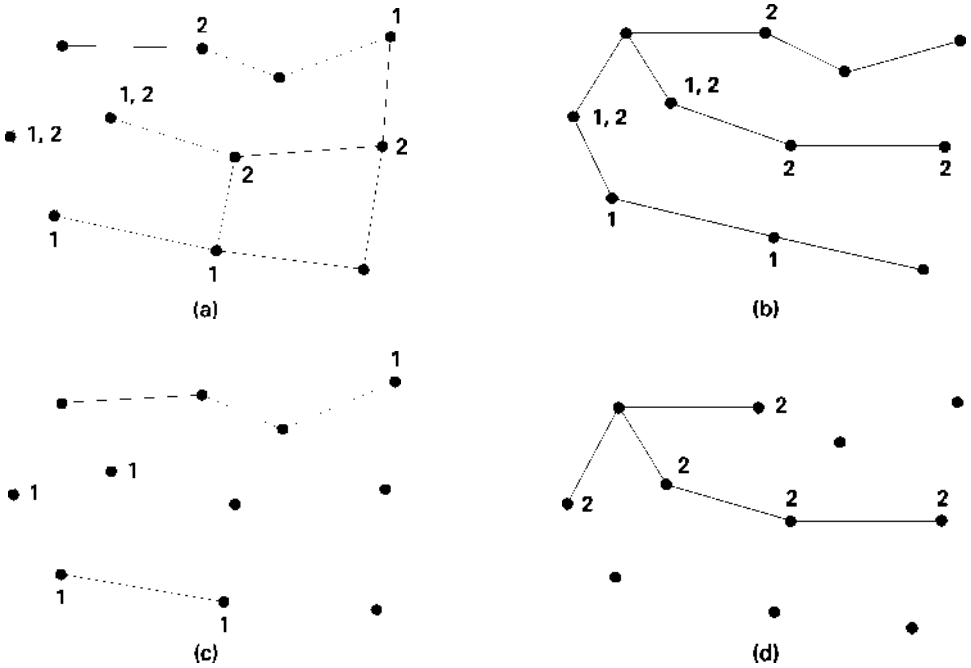


Fig. 5-17. (a) O reţea. (b) Un arbore de acoperire pentru cel mai din stânga ruter. (c) Un arbore multicast al grupului 1. (d) Un arborele multicast al grupului 2.

Un potenţial dezavantaj al acestui algoritm este acela că se comportă deficitar în cazul reţelelor extinse. Să presupunem că o reţea are *n* grupuri, fiecare cu un număr mediu de *m* membri. Pentru fiecare grup, trebuie memoraţi *m* arbori de acoperire retezaţi, deci un total de *mn* arbori. Dacă există multe grupuri mari, atunci, pentru a memora toţi arborii, este necesar un spaţiu de memorie considerabil.

O alternativă de proiectare foloseşte **arbori de bază** (**core-base trees**) (Ballardie ş.a., 1993). Aici se calculează un singur arbore de acoperire pentru fiecare grup, având rădăcina (core – inima, nucleu) lângă mijlocul grupului. Pentru a trimite un mesaj multicast, un calculator gazdă trimite mesajul către rădăcină, care apoi îl trimite de-a lungul arborelui de acoperire. Deşi acest arbore nu va fi optim pentru toate sursele, reducerea costului memorării de la *m* arbori la unul singur pe grup reprezintă o economie semnificativă.

5.2.9 Dirijarea pentru calculatoare gazdă mobile

Milioane de oameni deţin astăzi calculatoare portabile şi, de obicei, doresc să-şi citească poşta electronică şi să-şi acceseze sistemele de fişiere uzuale din orice punct al lumii s-ar afla. Aceste calculatoare mobile introduc o nouă complicaţie: pentru a dirija un pachet către un calculator mobil, reţeaua trebuie mai întâi să-l localizeze. Problema integrării calculatoarelor mobile într-o reţea este foarte recentă, dar în această secţiune vom sugera unele abordări şi vom da o posibilă soluţie.

Modelul lumii folosit de obicei de proiectanţii de reţele este cel prezentat în fig. 5-18. Aici avem o reţea WAN formată din rutere şi calculatoare gazdă. La WAN sunt conectate LAN-uri, MAN-uri şi celule de comunicaţie fără fir de tipul celor descrise în Cap. 2.

Calculatoarele gazdă care nu se mişcă niciodată se numesc **staţionare**. Ele sunt conectate la reţea prin fire de cupru sau fibre optice. Prin contrast, putem distinge alte două categorii de calculatoare gazdă. Calculatoarele gazdă **migratoare** sunt de fapt calculatoare gazdă staţionare, dar care, din când în când, se mută dintr-un loc fixat în altul şi care folosesc reţeaua doar atunci când sunt conectate fizic. Calculatoarele gazdă **călătoare** efectuează calcule în timp ce se deplasează şi doresc să menţină legătura în timpul deplasării. Vom folosi termenul **gazdă mobilă** pentru a desemna fiecare dintre ultimele două categorii, adică toate calculatoarele gazdă care sunt departe de casă şi doresc totuşi să fie conectate.

Se presupune că toate calculatoarele gazdă au o **locaţie de domiciliu** permanentă (home location), care nu se modifică niciodată. Calculatoarele gazdă au de asemenea o adresă personală permanentă, care poate fi folosită pentru a determina locaţia de domiciliu, într-o manieră similară celei în care numărul de telefon l-212-5551212 indică Statele Unite (codul de ţară 1) şi Manhattan (212). Scopul dirijării în sistemele cu calculatoare gazdă mobile este de a face posibilă trimiterea pachetelor spre gazde mobile folosind adresa lor personală permanentă şi să se asigure o trimitere eficientă a pachetelor spre ele, oriunde ar fi acestea. Problema este, bineînţeles, modul de localizare a lor.

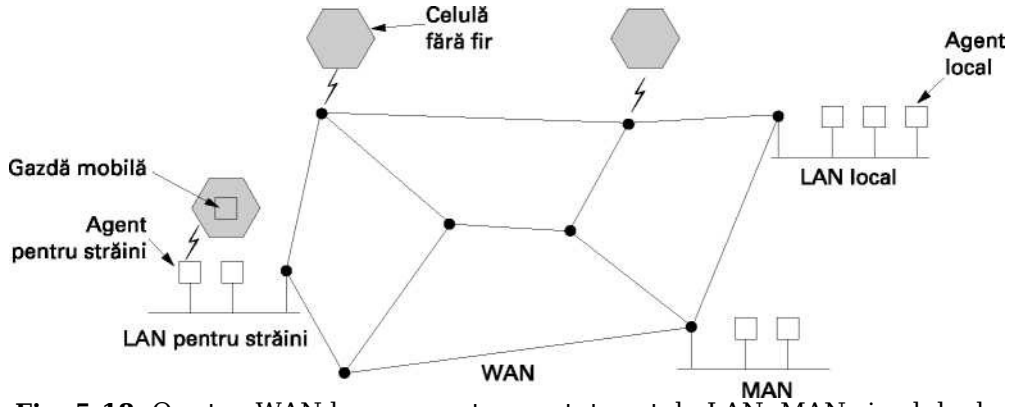


Fig. 5-18. O reţea WAN la care sunt conectate reţele LAN, MAN şi celule de comunicaţie fără fir.

În modelul din fig. 5-18, lumea este divizată (geografic) în unităţi de dimensiune redusă. Să denumim aceste unităţi domenii, unde un domeniu este de obicei un LAN sau o celulă de comunicaţie fără fir. Fiecare domeniu are unul sau mai mulţi **agenţi pentru străini (foreign agent)**, procese ce ţin evidenţa tuturor gazdelor mobile care vizitează domeniul. În plus, fiecare domeniu are un **agent local (local agent)** care ţine evidenţa calculatoarelor gazdă cu domiciliul în domeniul respectiv, dar care momentan vizitează alte domenii.

Când un nou calculator gazdă pătrunde într-un domeniu, fie prin conectarea la

acesta (ataşarea fizică la LAN), fie prin plimbarea printr-o celulă, trebuie să se înregistreze la agentul pentru străini din domeniul respectiv. Procedura de înregistrare se desfăşoară de obicei astfel:

1. Periodic, fiecare agent pentru străini difuzează un pachet anunţându-şi existenţa şi adresa. Un utilizator mobil nou sosit trebuie să aştepte unul dintre aceste mesaje, însă dacă nici unul nu soseşte într-un interval rezonabil de timp, calculatorul mobil poate difuza un pachet care spune: „Este vreun agent pentru străini prin zonă?”
2. Calculatorul mobil se înregistrează la agentul pentru străini precizând adresa sa permanentă, adresa actuală a nivelului legătură de date, precum şi unele informaţii de securitate.
3. Agentul pentru străini contactează apoi agentul local al domeniului din care provine utilizatorul mobil şi îi spune: „Unul dintre calculatoarele tale gazdă se află chiar aici.” Mesajul agentului pentru străini către agentul local conţine adresa de reţea a agentului pentru străini. El mai conţine de asemenea şi informaţia de securitate, pentru a convinge agentul local că gazda mobilă este într-adevăr acolo.
4. Agentul local examinează informaţia de securitate, care conţine şi o ştampilă de timp, pentru a dovedi că a fost generată în ultimele câteva secunde. Dacă este mulţumit, atunci anunţă agentul pentru străini că poate trece la acţiune.
5. Când agentul pentru străini primeşte confirmarea de la agentul local, el creează o nouă intrare în tabela sa şi anunţă utilizatorul mobil că a fost înregistrat.

Ideal, atunci când un calculator gazdă părăseşte un domeniu, acest lucru trebuie anunţat, pentru a fi scos din evidenţă, însă mulţi utilizatori pur şi simplu închid calculatorul când termină.

Când se trimite un pachet către o gazdă mobilă, el este dirijat către domiciliul calculatorului gazdă, deoarece adresa sugerează că aşa trebuie făcut, după cum se ilustrează şi în pasul 1 din fig. 5-19. Aici emiţătorul, aflat în oraşul de nord-vest Seattle, doreşte să trimită un pachet către un calculator gazdă aflat în New York, traversând Statele Unite. Pachetele trimise către calculatorul gazdă mobil în LAN-ul său de domiciliu, în New York, sunt interceptate de agentul local de acolo. Apoi agentul local caută noua locaţie (temporară) a gazdei mobile şi află adresa agentului pentru străini care se ocupă de gazdele mobile, în Los Angeles.

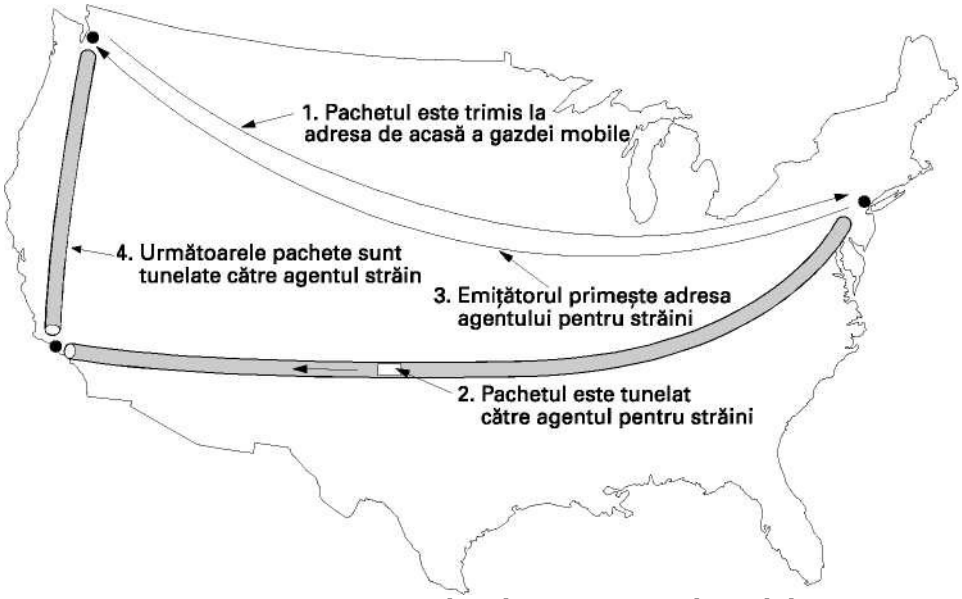

1. Pachetul este trimis la adresa de acasă a gazdei mobile
4. Următoarele pachete sunt tunelate către agentul străin
3. Emițătorul primește adresa agentului pentru străini
2. Pachetul este tunelat către agentul pentru străini

Fig. 5-19. Dirijarea pachetelor pentru gazde mobile.

Agentul local face două lucruri. În primul rând, încapsulează pachetul în câmpul de informaţie utilă (payload) al unui pachet de trimis, pe care îl expediază apoi către agentul pentru străini (pasul 2
din fig. 5-19). Acest mecanism se numeşte tunelare; îl vom studia în detaliu puţin mai târziu. După ce recepţionează pachetul încapsulat, agentul pentru străini extrage pachetul iniţial din câmpul payload şi îl trimite către calculatorul gazdă mobil drept cadru al nivelului legătură de date.

În al doilea rând, agentul local anunţă expeditorul mesajului ca de acum încolo să trimită pachetele adresate gazdei mobile încapsulându-le în câmpul de informaţie utilă (payload) al unor pachete trimise explicit agentului pentru străini, în loc să le trimită la adresa de domiciliu a calculatorului gazdă mobil (pasul 3). Pachetele următoare vor putea fi dirijate direct către utilizator, prin intermediul agentului pentru străini (pasul 4), evitând trecerea prin locaţia de domiciliu.

Diversele scheme propuse diferă prin mai multe aspecte. În primul rând, cât din acest protocol este realizat de rutere şi cât de calculatoarele gazdă şi, în această ultimă situaţie, pe care nivel de pe calculatorul gazdă. În al doilea rând, în unele scheme există rutere de-a lungul traseului care înregistrează adresele modificate, astfel încât să fie posibilă interceptarea şi redirectarea traficului chiar înainte ca acesta să ajungă la locaţia de domiciliu. În al treilea rând, în unele variante fiecare vizitator primeşte o adresă temporară unică; în altele adresa temporară se referă la un agent care se ocupă de toţi vizitatorii.

În al patrulea rând, variantele propuse diferă prin modul de rezolvare a situaţiei în care pachetele adresate unei destinaţii trebuie livrate în altă parte. O variantă este schimbarea adresei destinaţie şi retransmiterea pachetului modificat. Ca o alternativă, întregul pachet, adresa de domiciliu şi toate celelalte pot fi încapsulate în câmpul de informaţie utilă (payload) al altui pachet care este trimis spre adresa

temporară. În fine, schemele diferă prin aspectele legate de securitate. În general, când un ruter sau calculator gazdă primeşte un mesaj de forma „Începând din acest moment, vă rog să trimiteţi toate mesajele de poştă electronică ale lui Stephany către mine,” el ar putea avea dubii în legătură cu partenerul de dialog şi dacă este o idee bună. Diferite protocoale pentru calculatoare gazdă mobile sunt discutate şi comparate în (Hac şi Guo, 2000; Perkins, 1998a; Snoeren şi Balakrishan, 2000; Solomon, 1998; şi Wang şi Chen, 2001).

5.2.10 Dirijarea în reţele AD HOC

Am văzut cum se face dirijarea atunci când calculatoarele gazdă sunt mobile, dar ruterele sunt fixe. Un caz aparte apare atunci când însăşi ruterele sunt mobile. Printre situaţiile posibile sunt:

1. Vehicule militare pe un câmp de luptă atunci când nu există o infrastructură.
2. O flotă aflată în larg.
3. Echipe de intervenţie la un cutremur ce a distrus infrastructura.
4. O adunare de persoane cu calculatoare portabile într-o zonă fără 802.11.

În toate aceste cazuri, şi altele, fiecare nod este compus dintr-un ruter şi o gazdă, de obicei pe acelaşi calculator. Reţelele de noduri ce întâmplător se află aproape unul de celălalt sunt numite **reţele ad hoc** (**ad hoc network**) sau **MANET** (**Mobile Ad hoc NETworks**, reţele mobile ad hoc). Să le studiem pe scurt. Mai multe informaţii pot fi găsite în (Perkins, 2001).

Diferenţa între reţelele ad hoc şi reţelele cablate (eng.: wired) este că toate regulile despre topologii fixe, vecini ficşi şi cunoscuţi, relaţii fixe stabilite între adresa IP şi locaţie, şi altele, sunt complet suprimate. Cât ai clipi ruterele pot să apară şi să dispară sau să apară în alte locuri. În cazul reţelelor cablate, dacă un ruter are o cale validă către o destinaţie, aceasta continuă sa fie validă un timp nedefinit (suportând o defecţiune în altă parte a sistemului). Cu o reţea ad hoc, topologia se poate schimba mereu, aşa că oportunitatea şi chiar validitatea cailor se poate schimba spontan, fără avertizare. Nu mai trebuie spus că în aceste circumstanţe dirijarea în reţelele ad hoc diferă foarte mult faţă de dirijarea în echivalentul lor fix.

Au fost propuşi diferiţi algoritmi de dirijare pentru reţelele ad hoc. Unul dintre cei mai interesanţi este algoritmul de dirijare **AODV** (**Ad hoc On-demand Distance Vector**, vectori distanţă ad hoc la cerere) (Perkins şi Royer, 1999). Este o rudă îndepărtată a algoritmului cu vectori distanţă BellmanFord dar adaptat pentru un mediu mobil şi care ia în considerare limitele lărgimii de bandă şi durata scurtă de funcţionare a unei baterii în acest mediu. O altă caracteristică neobişnuită este faptul că acesta este un algoritm la cerere, adică determină o cale către o anumită destinaţie doar atunci când cineva doreşte să trimită un pachet către acea destinaţie. Să vedem acum ce înseamnă asta.

Descoperirea rutei

La orice moment de timp, o reţea ad hoc poate fi descrisă de un graf de noduri

(rutere + gazde). Două noduri sunt conectate (de exemplu, există un arc ce le uneşte în graf) dacă pot comunica direct folosind radioul. Cum unul dintre ele poate avea un transmiţător mai puternic decât celălalt, este posibil ca *A* să fie conectat cu *B* dar *B* să nu fie conectat cu *A.* Totuşi, pentru a simplifica, presupunem că toate conexiunile sunt simetrice. De asemenea ar trebui observat că simplul fapt că fiecare din cele două noduri este în raza radio a celuilalt nu înseamnă că ele sunt conectate. Pot exista clădiri, dealuri sau alte obstacole care să blocheze comunicaţia.

Pentru a descrie algoritmul, considerăm reţeaua ad hoc din fig. 5-20, în care un proces al nodului *A* doreşte să transmită un pachet nodului *I.* Algoritmul AODV menţine o tabelă în fiecare nod, în care cheia este destinaţia, şi care dă informaţii despre acea destinaţie, inclusiv cărui vecin trebuie trimise pachetele pentru a ajunge la destinaţie. Să presupunem că *A* se uită în tabelă şi nu găseşte nici o intrare pentru *I.* Trebuie deci să descopere o rută până la *I.* Această proprietate de descoperire a rutelor doar atunci când sunt necesare este cea care face din acest algoritm un algoritm „la cerere”.

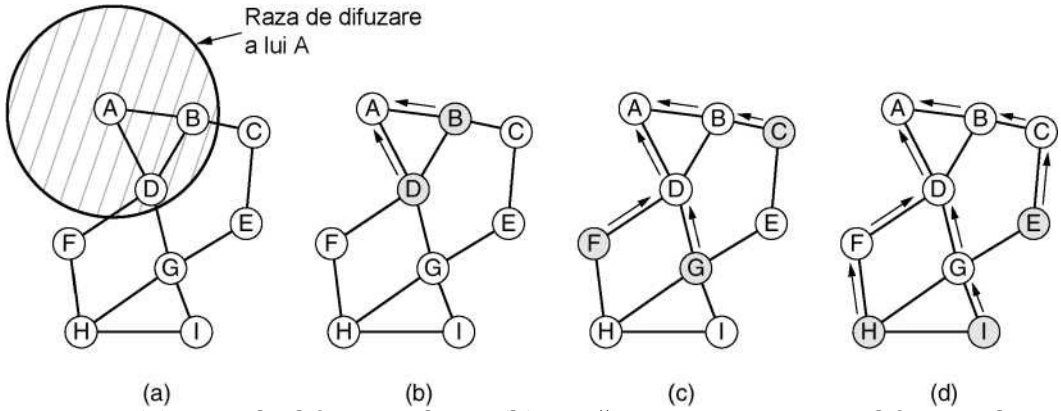


Fig. 5-20. (a) Raza de difuzare a lui *A*. (b) După ce *B* şi *D* au primit difuzarea lui *A*. (c) După ce *C, F* şi *G* au primit difuzarea lui *A*. (d) După ce *E, H* şi *I* au primit difuzarea lui *A*. Nodurile haşurate sunt noii destinatari. Săgeţile arată calea inversă posibilă.

Pentru a îl localiza pe *I, A* construieşte un pachet special ROUTE REQUEST şi îl difuzează. Pachetul ajunge la *B* şi la *D,* aşa cum este ilustrat în fig. 5-20(a). De fapt, motivul pentru care *B* şi *D* sunt conectate cu *A* în graf este acela că pot comunica cu *A*. *F,* de exemplu, nu este prezentat ca fiind legat printr-un arc cu *A* deoarece nu poate recepţiona semnalul radio emis de *A*.

Formatul pachetului ROUTE REQUEST este prezentat în fig. 5-21. El conţine adresele sursă şi destinaţie, bineînţeles adrese IP, care specifică cine caută pe cine. Conţine de asemenea şi un *ID Cerere* (Request ID), care este un contor local ţinut de fiecare nod şi incrementat la fiecare difuzare a unui pachet ROUTE REQUEST. Împreună, câmpurile *adresă* Sursă şi *ID Cerere* identifică unic pachetul ROUTE REQUEST pentru a permite nodurilor să înlăture orice duplicat pe care-l primesc.

Adresa sursei	ID cerere	Adresa destinaţiei	Numărul de secvenţă al sursei	Numărul de secvenţă al destinaţiei	Contorul de salturi

Fig. 5-21. Formatul unui pachet ROUTE REQUEST

În plus, pe lângă contorul *ID Cerere,* fiecare nod menţine un al doilea contor de secvenţă incrementat de fiecare dată când este trimis un pachet ROUTE REQUEST (sau un răspuns la un ROUTE REQUEST). Acesta funcţionează asemănător unui ceas şi este folosit pentru a deosebi rutele noi de cele vechi. Cel de-al patrulea câmp din fig. 5-21. este contorul de secvenţe al lui *A;* al cincilea câmp este cea mai recentă valoare a contorului de secvenţe a lui *I* întâlnită de *A* (0 dacă nu a întâlnit-o niciodată). Rolul acestor câmpuri se va clarifica în scurt timp. Ultimul câmp, *contorul de salturi* (Hop count), va memora câte salturi a făcut pachetul. Acesta este iniţializat cu 0.

Când un pachet ROUTE REQUEST ajunge la un nod (în acest caz *B* şi *D*), este prelucrat astfel:

1. Perechea (Adresă sursă, ID Cerere) este căutată într-o tabelă locală cu istoria cererilor pentru a vedea dacă această cerere a mai fost întâlnită şi procesată. Dacă este un duplicat, atunci este înlăturat şi procesarea se opreşte. Dacă nu este un duplicat, atunci perechea este introdusă tabelă pentru ca viitoarele duplicate să poată fi rejectate, şi procesarea continuă.
2. Receptorul caută destinaţia în propria tabela de dirijare. Dacă a aflat o nouă cale către destinaţie, atunci sursei îi este trimis un pachet ROUTE REPLY, spunându-i cum se ajunge la destinaţie. Nouă înseamnă că *numărul de secvenţă al destinaţiei* memorat în tabela de dirijare este mai mare sau egal cu numărul de secvenţă al destinaţiei din pachetul ROUTE REQUEST. Dacă este mai mic, calea memorată este mai veche decât calea anterioară pe care sursa o avea pentru destinaţie, aşa că se execută pasul 3.
3. Deoarece receptorul nu ştie o rută nouă către destinaţie, incrementează câmpul *contor de salturi* şi redifuzează pachetul ROUTE REQUEST. De asemenea extrage datele din pachet şi le memorează ca o nouă intrare în tabela de căi inverse. Această informaţie va fi folosită la construcţia caii inverse pentru ca răspunsul să se poată întoarce mai târziu la sursă. Săgeţile din fig. 5-20 sunt folosite pentru construirea căii inverse. De asemenea este pornit un cronometru asociat intrării corespunzătoare căii inverse nou create. Dacă expiră, intrarea este ştearsă.

Nici B şi nici D nu ştiu unde este I, aşa că fiecare creează o intrare pentru calea inversă indicând către A, aşa cum arată săgeţile din fig. 5-20, şi difuzează pachetul cu contorul de salturi setat pe 1. Difuzarea de la B ajunge la C şi D. C creează o intrare pentru el în tabela de căi inverse şi-l redifuzează. În contrast, *D* îl rejectează ca pe un duplicat. Similar, difuzarea făcută de *D* este rejectată de *B*. Totuşi, difuzarea făcută de *D* este acceptată de *F* şi *G* şi memorată, aşa cum este prezentat în fig. 5-20(c). După ce *E, H* şi *I* primesc difuzarea, pachetul ROUTE REQUEST ajunge în final la o destinaţie care ştie unde se află *I*, de fapt, chiar *I*, aşa cum este

ilustrat în fig. 5-20(d). Observaţi că deşi am prezentat difuzările în trei paşi discreţi, difuzările de la noduri diferite nu sunt coordonate în nici un fel.

Ca răspuns la cererea venită, *I* construieşte un pachet ROUTE REPLY, prezentat în fig. 5-22. *Adresa sursă, Adresa destinaţie* şi *Contorul de salturi* sunt copiate din cererea venită, dar *Numărul de secvenţă al destinaţiei e*ste luat din contorul său din memorie. Contorul de salturi este setat la 0. Câmpul *Durată de viaţă* controlează perioada în care ruta este validă. Acest pachet este trimis doar către nodul de la care a venit pachetul ROUTE REQUEST, în acest caz, *G.* Apoi urmează calea inversă către *D* şi în final către *A*. La fiecare nod, *Contorul de salturi* este incrementat astfel încât nodul să poată vedea cât de departe de destinaţie (*I*) este.

Adresa sursei	Adresa destinaţiei	Numărul de sec-venţă al destinaţiei	Contorul de salturi	Durata de viaţă

Fig. 5-22. Formatul pachetului ROUTE REPLY

Pachetul este analizat la fiecare nod intermediar de pe drumul înapoi. El este introdus în tabela locală de dirijare ca rută către *I* dacă una sau mai multe din cele trei condiţii de mai jos este îndeplinită:

1. Nu se cunoaşte nici o rută către *I*.
2. Numărul de secvenţă pentru *I* din pachetul ROUTE REPLY este mai mare decât valoarea din tabela de dirijare.
3. Numerele de secvenţă sunt egale, dar noua rută este mai scurtă.

În acest mod, toate nodurile de pe calea inversă află şi ruta către *I*, ca o consecinţă a descoperirii rutei de către *A.* Nodurile care au primit pachetul ROUTE REQUEST original, dar nu sunt pe calea inversă *(B, C, E, F,* şi *H* în acest exemplu) vor înlătura intrarea din tabela de căi inverse la expirarea timpului asociat.

Într-o reţea mare, algoritmul generează multe difuzări, chiar pentru destinaţii aflate foarte aproape. Numărul de difuzări poate fi redus după cum urmează. *Durata de viaţă* a pachetului IP este iniţializată de emiţător cu valoarea diametrului reţelei şi decrementată la fiecare salt. Dacă ajunge la valoarea 0, pachetul este înlăturat în loc să fie difuzat.

Procesul descoperirii este modificat după cum urmează. Pentru a localiza o destinaţie, emiţătorul difuzează un pachet ROUTE REQUEST cu *Durata de viaţă* setată la 1. Dacă nu primeşte nici un răspuns într-un timp rezonabil, este trimis un alt pachet, de această dată cu *Durata de viaţă* setată la 2. La încercările următoare se utilizează 3, 4, 5 etc. În acest fel se încearcă întâi local, apoi în cercuri din ce în ce mai largi.

Întreţinerea rutei

Deoarece nodurile se pot muta sau pot fi oprite, topologia se poate schimba spontan. De exemplu, în fig. 5-20, dacă *G* este oprit, *A* nu îşi va da seama că ruta pe care o folosea către *I (ADGI)* nu mai este validă . Algoritmul trebuie să poată rezolva această situaţie. Periodic, fiecare nod difuzează un mesaj *Hello.* Fiecare vecin ar trebui să răspundă . Dacă nu este primit nici un răspuns, atunci emiţătorul ştie că respectivul vecin s-a mutat din raza sa şi nu mai este conectat cu el. Similar,

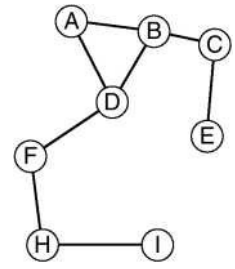
dacă încearcă să-i trimită un pachet unui vecin ce nu răspunde, află că vecinul nu mai este disponibil.

Această informaţie este folosită pentru a elimina rutele care nu mai funcţionează. Pentru fiecare destinaţie posibilă, fiecare nod, N, memorează vecinii care au trimis un pachet către acea destinaţie în ultimele AT secunde. Aceştia sunt numiţi **vecinii activi** (**active neighbors**) ai lui N . N realizează acest lucru prin intermediul unei tabele de dirijare în care cheia este destinaţia şi care conţine nodul de ieşire ce trebuie utilizat pentru a ajunge la destinaţie, contorul de salturi până la destinaţie, cel mai recent

număr de secvenţă al destinaţiei şi lista vecinilor activi pentru acea destinaţie. O tabelă de dirijare posibilă pentru nodul *D* pentru topologia din exemplul nostru este prezentată în fig. 5-23(a).

Dest.	Următorul salt	Distanţa	Vecini activi	Alte câmpuri
A	A	1	F, G	
B	B	1	F, G	
C	B	2	F	
E	G	2		
F	F	1	A, B	
G	G	1	A, B	
H	F	2	A, B	
I	G	2	A, B	

(a)



(b)

Fig. 5-23. (a) Tabela de dirijare a lui *D* înaintea opririi lui *G.* (b) Graful

Când oricare dintre vecinii lui *N* devine inaccesibil, el verifică tabela de dirijare pentru a vedea ce destinaţii au rute ce trec prin vecinul dispărut. Pentru fiecare din aceste rute, vecinii activi sunt informaţi că ruta ce trece prin *N* este acum inutilizabilă şi trebuie ştearsă din tabela lor de dirijare. Vecinii activi transmit mai departe informaţia vecinilor lor activi, şi aşa mai departe, recursiv, până ce toate rutele ce depindeau de nodul devenit inaccesibil sunt şterse din toate tabelele de dirijare.

Ca exemplu de întreţinere a rutelor, să considerăm exemplul anterior, dar cu *G* oprit brusc. Topologia schimbată este ilustrată în fig. 5-23(b). Când *D* descoperă că *G* nu mai este în reţea, se uită în tabela de dirijare şi observă că *G* era folosit pentru rutele către *E, G,* şi *I.* Mulţimea vecinilor activi pentru aceste destinaţii este *{A, B}.* Cu alte cuvinte, *A* şi *B* depind de *G* pentru câteva dintre rutele lor, aşa că ele trebuie informate că aceste rute nu mai sunt valide. *D* le informează prin trimiterea de pachete care le determină să-şi actualizeze tabelele de dirijare. De asemenea, *D* şterge intrările pentru *E, G,* şi *I* din tabela sa de dirijare.

Poate nu a fost evident din descrierea noastră, dar o diferenţă critică între AODV şi BellmanFord este că nodurile nu difuzează periodic întreaga lor tabelă de dirijare. Această diferenţă economiseşte atât lăţimea de bandă, cât şi durata de viaţă a bateriei.

De asemenea, AODV este capabil de dirijare prin difuzare şi de dirijare cu trimitere multiplă. Pentru detalii, consultaţi (Perkins şi Royer, 2001). Dirijarea ad hoc este un domeniu de cercetare fierbinte. S-a publicat mult despre acest subiect. Câteva dintre lucrări sunt (Chen et. al, 2002; Hu şi Johnson, 2001; Li et. al, 2001; Raju şi Garcia-Luna-Aceves, 2001; Ramanathan şi Redi, 2002; Royer şi Toh, 1999; Spohn şi Garcia-Luna-Aceves, 2001; Tseng et. al, 2001; şi Zadeh et. al, 2002).

5.2.11 Căutarea nodurilor în reţele punct la punct

Un fenomen relativ nou este reprezentat de reţelele punct la punct, în care un număr mare de persoane, de obicei cu conexiuni permanente la Internet, sunt în contact pentru a partaja resurse. Prima aplicaţie larg răspândită a tehnologiei punct la punct a fost un delict de anvergură: 50 de milioane de utilizatori ai Napster-ului făceau schimb de cântece cu copyright fără permisiunea proprietarilor până când utilizarea Napster-ului a fost interzisă de tribunal în vâltoarea unor mari controverse. Totuşi, tehnologia punct la punct are multe utilizări interesante şi legale. De asemenea, are ceva asemănător cu problema dirijării, deşi nu este chiar la fel cu cele studiate până acum. Cu toate acestea, merită să-i acordăm puţină atenţie.

Ceea ce face ca sistemele punct la punct să fie interesante este faptul că ele sunt în întregime distribuite. Toate nodurile sunt simetrice şi nu există un control central sau o ierarhie. Într-un sistem punct la punct tipic fiecare utilizator are o informaţie ce poate fi de interes pentru alţi utilizatori. Această informaţie poate fi software gratuit, muzică (din domeniul public), fotografii şi aşa mai departe. Dacă numărul de utilizatori este mare, aceştia nu se ştiu între ei şi nu ştiu unde să găsească ceea ce caută. O soluţie este o bază de date centrală, dar aşa ceva ar putea fi irealizabil din diverse motive (de exemplu, nimeni nu doreşte să o găzduiască şi s-o administreze). Astfel, problema se reduce la felul în care un utilizator găseşte un nod ce conţine ceea ce caută el, în absenţa unei baze de date centralizate sau a unui index centralizat.

Să presupunem că un utilizator are una sau mai multe unităţi de date cum ar fi cântece, fotografii, programe, fişiere ş.a.m.d, pe care ar dori să le citească alţi utilizatori. Fiecare unitate are ca nume un şir de caractere ASCII. Un posibil utilizator cunoaşte numai şirul de caractere ASCII şi vrea să afle dacă una sau mai multe persoane au copii, şi în caz că au, care sunt adresele IP ale acestora.

De exemplu, să considerăm o bază de date genealogică distribuită. Fiecare specialist în genealogie are o serie de înregistrări on-line despre strămoşii şi rudele sale, posibil şi cu fotografii, clipuri audio sau chiar video ale persoanei. Mai multe persoane pot avea acelaşi străbunic, astfel că un strămoş poate avea înregistrări în mai multe noduri. Numele înregistrării este numele persoanei într- o formă canonică. La un moment dat, un specialist în genealogie descoperă testamentul străbunicului său într-o arhivă, în care străbunicul lasă moştenire nepotului ceasul de buzunar de aur. Genea- logistul ştie numele nepotului şi doreşte să afle dacă alt genealogist are o înregistrare pentru el. Cum aflăm, fără o bază de date centrală, cine are, dacă are cineva, aceste înregistrări?

Pentru a rezolva această problemă au fost propuşi diverşi algoritmi. Cel pe care îl vom studia este Chord (Dabek ş.a.,2001; şi Stoica ş.a.,2001). O explicaţie simplificată a felului în care funcţionează este următoarea. Sistemul Chord constă din n utilizatori participanţi, fiecare dintre ei având câteva înregistrări memorate şi fiecare dintre ei fiind pregătit să memoreze fragmente din index pentru uzul celorlalţi utilizatori. Fiecare nod utilizator are o adresă IP care poate fi convertită într-un număr pe m biţi, folosind o funcţie de dispersie, *hash.* Pentru *hash* Chord

foloseşte SHA-1. SHA-l este folosit în criptografie; îl vom analiza în Cap. 8. Pentru moment, este doar o funcţie care primeşte ca argument un şir de caractere de lungime variabilă şi generează un număr aleator pe 160 de biţi. Deci, putem converti orice adresă IP într-un număr pe 160 de biţi numit **identificatorul nodului** (**node identifier**).

Conceptual, toţi cei 2^{160} identificatori de noduri sunt aşezaţi în ordine crescătoare într-un mare cerc. Câţiva corespund nodurilor participante, dar majoritatea nu. În fig. 5-24(a) prezentăm un cerc de identificatori de noduri pentru $m = 5$ (pentru moment ignoraţi arcele din centru). În acest exemplu, nodurile cu identificatorii 1, 4, 7, 12, 15, 20 şi 27 corespund nodurilor reale şi sunt haşurate; restul nodurilor nu există.

Să definim acum funcţia *successor(k), ca* identificator al primului nod real ce urmează după *k* pe cerc, în sensul acelor de ceasornic. De exemplu, successor(6) = 7, successor(8) = 12 şi successor(22) = 27.

Numele înregistrărilor (nume de cântece, numele strămoşilor etc.) sunt de asemenea convertite cu funcţia *hash* (adică SHA-1) pentru a genera un număr pe 160 de biţi, numit **cheie (key)**. Astfel, pentru a converti un *nume* (numele ASCII al înregistrării) la cheia sa, folosim *key = hash (name).* Acest calcul este doar un apel local al procedurii *hash*. Dacă o persoană deţinând o înregistrare genealogică pentru *nume* doreşte să o facă publică, atunci construieşte mai întâi un tuplu *(nume, adresă IP)* şi apoi îl roagă pe *successor(hash(nume))* să-l memoreze. Dacă există mai multe înregistrări (în noduri diferite) pentru acelaşi nume, toate tuplurile lor vor fi memorate în acelaşi nod. În acest mod, index-ul este distribuit aleator pe noduri. Pentru toleranţa la defecte, pentru memorarea fiecărui tuplu în *p* noduri pot fi folosite *p* funcţii de dispersie diferite, dar nu ne vom referi aici la acest aspect.

Dacă ulterior un utilizator caută *nume,* îi aplică funcţia de dispersie pentru a afla cheia şi apoi foloseşte *successor(cheie)* pentru a afla adresa IP a nodului ce memorează tuplurile respective din index. Primul pas este uşor; dar al doilea nu. Pentru a face posibilă găsirea adresei IP corespunzătoare unei anumite chei, fiecare nod trebuie să menţină structuri de date administrative. Una dintre acestea este adresa IP a nodului succesor, de-a lungul cercului de identificatori de noduri. De exemplu, în fig. 5-24, succesorul nodului 4 este 7 şi succesorul nodului 7 este 12.

Căutarea se poate desfăşura după cum urmează. Nodul care face cererea trimite succesorului său un pachet conţinând adresa sa IP şi cheia pe care o caută. Pachetul se propagă prin inel până ce localizează succesorul identificatorului căutat. Nodul verifică dacă deţine vreo informaţie cu cheia respectivă şi, dacă este aşa, o returnează direct nodului care a făcut cererea, a cărui adresă IP o are.

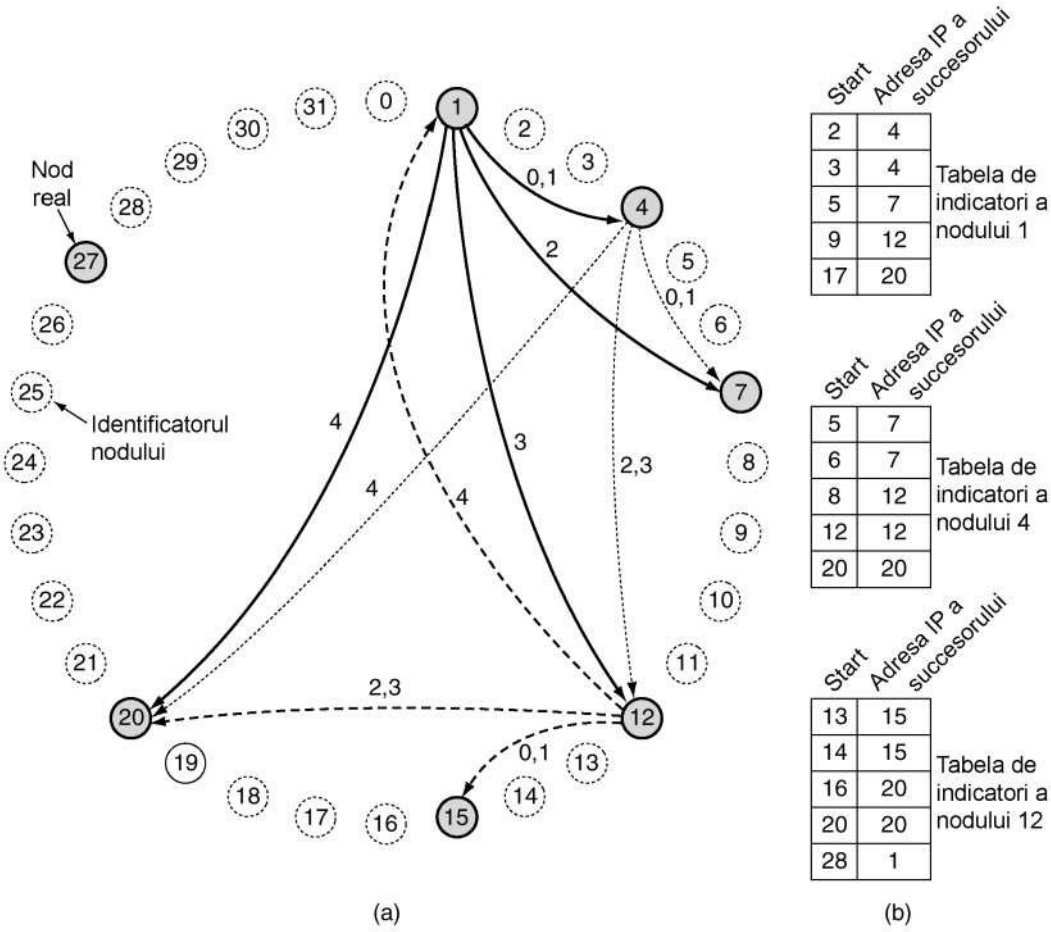


Fig. 5-24. (a) Un set de 32 de identificatori de noduri aranjaţi în cerc. Nodurile haşurate corespund maşinilor existente. Arcele arată indicatorii de la nodurile 1, 4 şi 12. Etichetele arcelor reprezintă indici în tabele. (b) Exemple de tabele de indicatori.

Ca o primă optimizare, fiecare nod ar putea memora adresele IP ale succesorului şi predecesorului, astfel încât interogările ar putea fi trimise fie în sensul acelor de ceasornic, fie în sens invers, pe calea considerată mai scurtă. De exemplu, nodul 7 din fig. 5-24 poate să trimită în sensul acelor de ceasornic pentru a găsi nodul cu identificatorul 10, dar în sens invers pentru a găsi nodul cu identificatorul 3.

Chiar şi cu două posibilităţi pentru direcţie, căutarea liniară prin toate nodurile este foarte ineficientă într-un sistem punct la punct mare, deoarece numărul mediu de noduri implicat într-o căutare este de *n/2.* Pentru a mări considerabil viteza de căutare, fiecare nod memorează de asemenea ceea ce Chord numeşte **tabelă de indicatori (finger table)**. Tabela de indicatori are *m* intrări, indexate de la 0 la *m-l,* fiecare indicând către un nod real diferit. Fiecare dintre intrări are două câmpuri: *start* şi adresa IP a *successor(start),* aşa cum este prezentat pentru cele trei noduri din fig. 5-24(b). Valoarea acestor câmpuri pentru intrarea *i* a nodului *k* sunt:

start = k + 2 (modulo 2^m)
Adresa IP a successor(start[i])

Observaţi că fiecare nod memorează adresa IP a unui număr relativ mic de noduri şi că majoritatea acestora sunt foarte apropiate din punctul de vedere al identificatorilor de noduri.

Folosind tabela de indicatori, căutarea *cheii* la nodul *k* se desfăşoară după cum urmează. Dacă *cheia* este între *k* şi *successor(k),* atunci nodul care deţine informaţia despre *cheie* este *succesor(k)* şi căutarea se termină. Altfel, se caută în tabela de indicatori pentru a găsi intrarea al cărui câmp *start* este predecesorul cel mai apropiat al *cheii.* Apoi se trimite direct la adresa IP din intrarea din tabela de indicatori o cerere în care se cere continuarea căutării. Deoarece aceasta este mai apropiată de cheie, dar inferioară, sunt mari şanse să fie capabilă să returneze un răspuns după un număr mic de interogări suplimentare. De fapt, deoarece fiecare căutare înjumătăţeşte distanţa rămasă până la ţintă, se poate demonstra că numărul mediu de căutări este $log_2 n$.

Ca prim exemplu, să considerăm căutarea *key* = 3 la nodul 1. Deoarece nodul 1 ştie că 3 se află între el şi succesorul său, 4, nodul dorit este 4 şi căutarea s-a terminat, returnându-se adresa IP a nodului 4.

Ca un al doilea exemplu, să considerăm căutarea *key* = 14 la nodul 1. Deoarece 14 nu este între 1 şi 4, este analizată tabela de indicatori. Cel mai apropiat predecesor al lui 14 este 9, aşa că cererea este trimisă către adresa IP din intrarea 9, şi anume, cea a nodului 12. Nodul 12 vede că 14 se află între el şi successor(15), aşa că returnează adresa IP a nodului 15.

Ca un al treilea exemplu, să considerăm căutarea *key* = 16 la nodul 1. Din nou se trimite o interogare la nodul 12, dar de această dată nodul 12 nu cunoaşte răspunsul. Caută nodul cel mai apropiat predecesor al lui 16 şi îl găseşte pe 14, care furnizează adresa IP a nodului 15. Acestuia îi este trimisă o interogare. Nodul 15 observă că 16 se află între el şi succesorul său (20), aşa că returnează apelantului adresa IP a lui 20, care se întoarce către nodul 1.

Deoarece nodurile apar şi dispar mereu, Chord trebuie să trateze cumva aceste operaţii. Presupunem că atunci când sistemul a început să funcţioneze, el era îndeajuns de mic pentru ca nodurile să poată schimba informaţii direct, pentru a construi primul cerc şi tabelele de indicatori. După aceea este necesară o procedură automată, după cum urmează. Când un nod nou, *r*, vrea să se alăture, trebuie să contacteze un nod existent şi să-i ceară să caute adresa IP a successor(r). După aceea noul nod îl întreabă pe successor(r) cine este predecesorul său. Apoi noul nod cere ambelor noduri să îl insereze pe *r* între ele în cerc. De exemplu, dacă 24 din fig. 5-24 vrea să se alăture, roagă orice nod să caute *successor(24),* care este 27. Apoi îl întreabă pe 27 cine este predecesorul său (20). După ce le înştiinţează pe amândouă de existenţa sa, 20 îl foloseşte pe 24 ca succesor, iar 27 îl foloseşte pe 24 ca predecesor. În plus, nodul 27 predă cheile din intervalul 21-24, care acum îi aparţin lui 24. În acest moment, 24 este inserat pe deplin.

Totuşi, multe tabele de indicatori sunt acum greşite. Pentru a le corecta, fiecare nod rulează un proces în fundal care calculează periodic fiecare indicator prin

apelarea funcţiei *successor.* Când una dintre aceste interogări ajunge la un nod nou, este actualizată intrarea corespunzătoare indicatorului.

Când un nod pleacă normal, predă cheile succesorului său şi informează predecesorul de plecarea sa, astfel încât predecesorul poate să se lege la succesorul nodului care a plecat. Când se defectează un nod apare o problemă, deoarece predecesorul său nu are un succesor valid. Pentru a soluţiona problema, fiecare nod păstrează date nu numai despre succesorul său direct, dar şi despre *s* succesori direcţi, pentru a-i permite să sară peste *s-1* noduri consecutive defecte şi să se reconecteze la cerc.

Chord a fost folosit la construirea unui sistem de fişiere distribuit (Dabek ş.a.,2001) şi alte aplicaţii, iar cercetările continuă. Un sistem punct la punct diferit, Pastry, şi aplicaţiile sale sunt descrise în (Rowstron şi Druschel, 2001a; şi Rowstron şi Druschel, 2001b). Un al treilea sistem punct la punct, Freenet, este discutat în (Clarke ş.a., 2002). Un al patrulea sistem de acest tip este descris în (Ratnasamy ş.a., 2001).

5.3 ALGORITMI PENTRU CONTROLUL CONGESTIEI

Atunci când foarte multe pachete sunt prezente într-o subreţea (sau o parte a unei subreţele), performanţele se degradează. Această situaţie se numeşte **congestie** (**congestion**). Fig. 5-25 prezintă simptomele. Când numărul de pachete emise în subreţea de calculatoarele gazdă nu depăşeşte capacitatea de transport, ele sunt livrate integral (cu excepţia celor care sunt afectate de erori de transmisie), iar numărul celor livrate este proporţional cu numărul celor emise. Totuşi, atunci când traficul creşte prea mult, ruterele încep să nu mai facă faţă şi să piardă pachete. Aceasta tinde să înrăutăţească lucrurile. La un trafic foarte intens performanţele se deteriorează complet şi aproape nici un pachet nu mai este livrat.

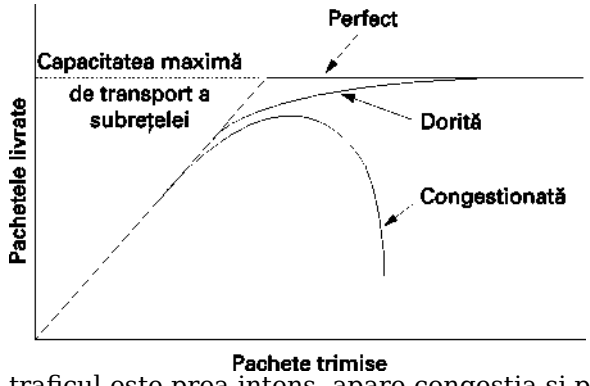


Fig. 5-25. Dacă traficul este prea intens, apare congestia şi performanţele se degradează puternic.

Congestia poate fi produsă de mai mulţi factori. Dacă dintr-o dată încep să sosească şiruri de pachete pe trei sau patru linii de intrare şi toate necesită aceeaşi linie de ieşire, atunci se va forma o coadă. Dacă nu există suficientă memorie pentru a le păstra pe toate, unele se vor pierde. Adăugarea de memorie poate fi folositoare până la un punct, dar Nagle (1987) a descoperit că dacă ruterele ar avea o cantitate infinită de memorie, congestia s-ar înrăutăţi în loc să se amelioreze, deoarece până să ajungă la începutul cozii pachetele au fost deja considerate pierdute (timeout repetat) şi s-au trimis duplicate. Toate aceste pachete vor fi trimise cu conştiinciozitate către următorul ruter, crescând încărcarea de-a lungul căii către destinaţie.

Şi procesoarele lente pot cauza congestia. Dacă unitatea centrală (CPU) a ruterului este lentă în execuţia funcţiilor sale (introducerea în cozi, actualizarea tabelelor etc.), se pot forma cozi, chiar dacă linia de comunicaţie nu este folosită la capacitate. Similar şi liniile cu lăţime de bandă scăzută pot provoca congestia. Schimbarea liniilor cu unele mai performante şi păstrarea aceluiaşi procesor sau vice-versa de obicei ajută puţin, însă de cele mai multe ori doar deplasează punctul critic. De asemenea, îmbunătăţirea parţială şi nu totală a sistemului cel mai adesea mută punctul critic în altă parte. Adevărata problemă este de multe ori o incompatibilitate între părţi ale sistemului. Ea va persista până ce toate componentele sunt în echilibru.

Este important să subliniem diferenţa dintre controlul congestiei şi controlul fluxului, deoarece relaţia este subtilă. Controlul congestiei trebuie să asigure că subreţeaua este capabilă să transporte întreg traficul implicat. Este o problemă globală, implicând comportamentul tuturor calculatoarelor gazdă, al tuturor ruterelor, prelucrarea de tip memorează-şi-retransmite (store-and-forward) din rutere şi toţi ceilalţi factori care tind să diminueze capacitatea de transport a subreţelei.

Controlul fluxului, prin contrast, se referă la traficul capăt la capăt între un expeditor şi un destinatar. Rolul său este de a împiedica un expeditor rapid să trimită date continuu, la o viteză mai mare decât cea cu care destinatarul poate consuma datele. Controlul fluxului implică frecvent existenţa unui feed-back de la receptor către emiţător, pentru a spune emiţătorului cum se desfăşoară lucrurile la celălalt capăt.

Pentru a vedea diferenţa dintre aceste două concepte, să considerăm o reţea cu fibre optice cu o capacitate de 1000 gigabiţi/sec pe care un supercalculator încearcă să transfere un fişier către un calculator personal la 1 Gbps. Deşi nu există nici o congestie (reţeaua nu are nici un fel de probleme), controlul fluxului este necesar pentru a forţa supercalculatorul să se oprească des, pentru a permite calculatorului personal să şi respire.

La cealaltă extremă, să considerăm o reţea de tip memorează-şi-retransmite (store-and-forward), cu linii de 1 Mbps şi 1000 de calculatoare mari, din care jumătate încearcă să transfere fişiere la 100 Kbps către cealaltă jumătate. Aici problema nu apare datorită unui emiţător rapid care surclasează un receptor lent, ci pentru că traficul cerut depăşeşte posibilităţile reţelei.

Motivul pentru care controlul congestiei şi controlul fluxului sunt adesea

confundate este acela că unii algoritmi pentru controlul congestiei funcţionează trimiţând mesaje înapoi către diferitele surse, spunându-le să încetinească atunci când reţeaua are probleme. Astfel, un calculator gazdă poate primi un mesaj de încetinire fie din cauză că receptorul nu suportă încărcarea, fie pentru că reţeaua este depăşită. Vom reveni asupra acestui punct mai târziu.

Vom începe studiul algoritmilor pentru controlul congestiei prin studiul unui model general de tratare a acesteia. Apoi ne vom ocupa de principalele măsuri pentru prevenirea sa. După aceea, vom analiza o serie de algoritmi dinamici pentru tratarea congestiei odată ce a apărut.

5.3.1 Principii generale ale controlului congestiei

Multe din problemele care apar în sistemele complexe, cum ar fi reţelele de calculatoare, pot fi privite din punctul de vedere al unei teorii a controlului. Această abordare conduce la împărţirea tuturor soluţiilor în două grupe: în buclă deschisă şi în buclă închisă. Soluţiile în buclă deschisă încearcă să rezolve problema printr-o proiectare atentă, în esenţă să se asigure că problema nu apare. După ce sistemul este pornit şi funcţionează, nu se mai fac nici un fel de corecţii.

Instrumentele pentru realizarea controlului în buclă deschisă decid când să se accepte trafic nou, când să se distrugă pachete şi care să fie acestea, realizează planificarea deciziilor în diferite puncte din reţea. Toate acestea au ca numitor comun faptul că iau decizii fără a ţine cont de starea curentă a reţelei.

Prin contrast, soluţiile în buclă închisă se bazează pe conceptul de reacţie inversă (feedback loop). Această abordare are trei părţi, atunci când se foloseşte pentru controlul congestiei:

1. Monitorizează sistemul pentru a detecta când şi unde se produce congestia.
2. Trimite aceste informaţii către locurile unde se pot executa acţiuni.
3. Ajustează funcţionarea sistemului pentru a corecta problema.

În vederea monitorizării subreţelei pentru congestie se pot folosi diverse de metrici. Cele mai utilizate sunt procentul din totalul pachetelor care au fost distruse din cauza lipsei spaţiului temporar de memorare, lungimea medie a cozilor de aşteptare, numărul de pachete care sunt retransmise pe motiv de timeout, întârzierea medie a unui pachet, deviaţia standard a întârzierii unui pachet. În toate cazurile, valorile crescătoare indică creşterea congestiei.

Al doilea pas în bucla de reacţie este transferul informaţiei legate de congestie de la punctul în care a fost depistată la punctul în care se poate face ceva. Varianta imediată presupune trimiterea unor pachete de la ruterul care a detectat congestia către sursa sau sursele de trafic, pentru a raporta problema. Evident, aceste pachete suplimentare cresc încărcarea reţelei exact la momentul în care acest lucru era cel mai puţin dorit, subreţeaua fiind congestionată.

Există însă şi alte posibilităţi. De exemplu, poate fi rezervat un bit sau un câmp în fiecare pachet, pentru a fi completat de rutere dacă congestia depăşeşte o anumită valoare de prag. Când un ruter detectează congestie, el completează câmpurile tuturor pachetelor expediate, pentru a-şi preveni vecinii.

O altă abordare este ca ruterele sau calculatoarele gazdă să trimită periodic

pachete de probă pentru a întreba explicit despre congestie. Aceste informaţii pot fi apoi folosite pentru a ocoli zonele cu probleme. Unele staţii de radio au elicoptere care zboară deasupra oraşelor pentru a raporta congestiile de pe drumuri şi a permite ascultătorilor mobili să îşi dirijeze pachetele (maşinile) astfel încât să ocolească zonele fierbinţi.

În toate schemele cu feedback se speră că informarea asupra producerii congestiei va determina calculatoarele gazdă să ia măsurile necesare pentru a reduce congestia. Pentru ca o schemă să funcţioneze corect, duratele trebuie reglate foarte atent. Dacă un ruter strigă STOP de fiecare dată când sosesc două pachete succesive şi PLEACĂ (eng.: GO) de fiecare dată când este liber mai mult de 20 psec, atunci sistemul va oscila puternic şi nu va converge niciodată. Pe de altă parte, dacă va aşteaptă 30 de minute pentru a fi sigur înainte de a spune ceva, mecanismul pentru controlul congestiei reacţionează prea lent pentru a fi de vreun folos real. Pentru a funcţiona corect sunt necesare unele medieri, dar aflarea celor mai potrivite constante de timp nu este o treabă tocmai uşoară.

Se cunosc numeroşi algoritmi pentru controlul congestiei. Pentru a oferi o modalitate de organizare a lor, Yang şi Reddy (1995) au dezvoltat o taxonomie pentru algoritmii de control al congestiei. Ei încep prin a împărţi algoritmii în cei în buclă deschisă şi cei în buclă închisă, aşa cum s-a precizat anterior. În continuare împart algoritmii cu buclă deschisă în unii care acţionează asupra sursei şi alţii care acţionează asupra destinaţiei. Algoritmii în buclă închisă sunt de asemenea împărţiţi în două subcategorii, cu feedback implicit şi cu feedback explicit. În algoritmii cu feedback explicit, pachetele sunt trimise înapoi de la punctul unde s-a produs congestia către sursă, pentru a o avertiza. În algoritmii impliciţi, sursa deduce existenţa congestiei din observaţii locale, cum ar fi timpul necesar pentru întoarcerea confirmărilor.

Prezenţa congestiei înseamnă că încărcarea (momentană) a sistemului este mai mare decât cea pe care o pot suporta resursele (unei părţi a sistemului). Pentru rezolvare vin imediat în minte două soluţii: sporirea resurselor sau reducerea încărcării. De exemplu subreţeaua poate începe să folosească linii telefonice pentru a creşte temporar lăţimea de bandă între anumite puncte. În sistemele bazate pe sateliţi, creşterea puterii de transmisie asigură de regulă creşterea lăţimii de bandă. Spargerea traficului pe mai multe căi în locul folosirii doar a celei mai bune poate duce efectiv la creşterea lăţimii de bandă. În fine, ruterele suplimentare, folosite de obicei doar ca rezerve pentru copii de siguranţă (backups) (pentru a face sistemul tolerant la defecte), pot fi folosite pentru a asigura o capacitate sporită atunci când apar congestii serioase.

Totuşi, uneori nu este posibilă creşterea capacităţii sau aceasta a fost deja crescută la limită. Atunci singura cale de a rezolva congestia este reducerea încărcării. Sunt posibile mai multe metode pentru reducerea încărcării, cum ar fi refuzul servirii anumitor utilizatori, degradarea serviciilor pentru o parte sau pentru toţi utilizatorii şi planificarea cererilor utilizatorilor într-o manieră mai previzibilă.

Unele dintre aceste metode, pe care le vom studia pe scurt, pot fi aplicate cel

mai bine circuitelor virtuale. Pentru subreţelele care folosesc intern circuite virtuale aceste metode pot fi utilizate la nivelul reţea. Pentru subreţele bazate pe datagrame ele pot fi totuşi folosite uneori pentru conexiuni la nivelul transport. În acest capitol, ne vom concentra pe folosirea lor în cadrul nivelului reţea. În următorul, vom vedea ce se poate face la nivelul transport pentru a controla congestia.

5.3.2 Politici pentru prevenirea congestiei

Să începem studiul asupra metodelor pentru controlul congestiei prin analiza sistemelor cu buclă deschisă. Aceste sisteme sunt proiectate astfel încât să minimizeze congestia, în loc să o lase să se producă şi apoi să reacţioneze. Ele încearcă să-şi atingă scopul folosind politici corespunzătoare, la diferite niveluri. În fig. 5-26 sunt prezentate diferite politici pentru nivelul legătură de date, reţea şi transport, care pot influenţa congestia (Jain, 1990).

Să începem cu nivelul legătură de date şi să ne continuăm apoi drumul spre niveluri superioare. Politica de retransmisie stabileşte cât de repede se produce timeout la un emiţător şi ce transmite acesta la producerea timeout-ului. Un emiţător vioi, care produce repede timeout şi retransmite toate pachetele în aşteptare folosind retrimiterea ultimelor *n,* va produce o încărcare mai mare decât un emiţător calm, care foloseşte retrimiterea selectivă. Strâns legată de acestea este politica de memorare în zonă tampon. Dacă receptorii distrug toate pachetele în afara secvenţei, acestea vor trebui retransmise ulterior, introducând o încărcare suplimentară. În ceea ce priveşte controlul congestiei, repetarea selectivă este, în mod sigur, mai bună decât retrimiterea ultimelor *n* .

Nivel	**Politică**
Transport	Politica de retransmisie Politica de memorare temporară a pachetelor în afară de secvenţă (out-of-order caching) Politica de confirmare Politica de control al fluxului Determinarea timeout-ului
Reţea	Circuite virtuale contra datagrame în interiorul subreţelei Plasarea pachetelor în cozi de aşteptare şi politici de servire Politica de distrugere a pachetelor Algoritmi de dirijare Gestiunea timpului de viaţă al pachetelor
Legătură de date	Politica de retransmitere Politica de memorare temporară a pachetelor în afară de secvenţă (out-of-order caching) Politica de confirmare Politica de control al fluxului

Fig. 5-26. Politici care influenţează congestia.

Şi politica de confirmare afectează congestia. Dacă fiecare pachet este confirmat imediat, pachetele de confirmare vor genera un trafic suplimentar. Totuşi, în cazul în care confirmările sunt preluate de traficul de răspuns, se pot produce timeout-uri şi retransmisii suplimentare. O schemă prea strânsă pentru controlul fluxului (de

exemplu fereastră mică) reduce volumul de date şi ajută în lupta cu congestia.

La nivelul reţea, alegerea între folosirea circuitelor virtuale şi datagrame influenţează congestia, deoarece mulţi algoritmi pentru controlul congestiei funcţionează doar pe subreţele cu circuite virtuale. Plasarea în cozi de aşteptare a pachetelor şi politicile de servire specifică dacă ruterele au o coadă pentru fiecare linie de intrare, o coadă pentru fiecare linie de ieşire sau ambele. Mai precizează ordinea în care se prelucrează pachetele (de exemplu round robin sau bazată pe priorităţi). Politica de distrugere a pachetelor este regula care stabileşte ce pachet este distrus dacă nu mai există spaţiu. O politică bună va ajuta la eliminarea congestiei, pe când una greşită o va accentua.

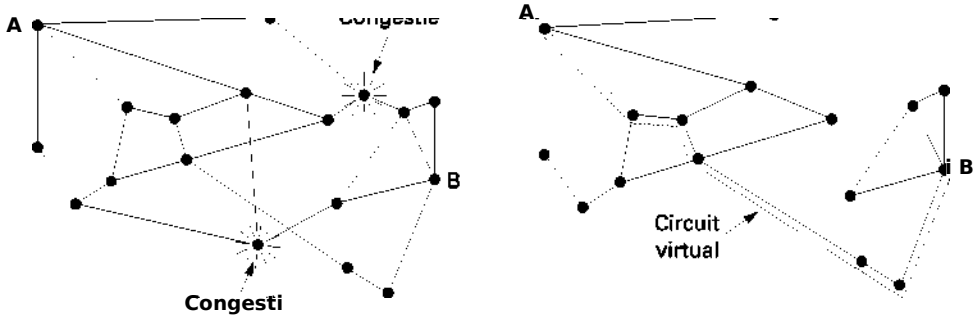
Un algoritm de dirijare bun poate ajuta la evitarea congestiei prin răspândirea traficului de-a lungul tuturor liniilor, în timp ce un algoritm neperformant ar putea trimite toate pachetele pe aceeaşi linie, care deja este congestionată. În fine, gestiunea timpului de viaţă asociat pachetelor stabileşte cât de mult poate trăi un pachet înainte de a fi distrus. Dacă acest timp este prea mare, pachetele pierdute vor încurca pentru mult timp activitatea, iar dacă este prea mic, este posibil să se producă timeout înainte de a ajunge la destinaţie, provocând astfel retransmisii.

La nivelul transport apar aceleaşi probleme ca la nivelul legăturii de date dar, în plus, determinarea intervalului de timeout este mai dificil de realizat, deoarece timpul de tranzit prin reţea este mai greu de prezis decât timpul de tranzit pe un fir între două rutere. Dacă intervalul de timeout este prea mic, vor fi trimise inutil pachete suplimentare. Dacă este prea mare, congestia se va reduce, însă timpul de răspuns va fi afectat de pierderea unui pachet.

5.3.3 Controlul congestiei în subreţelele bazate pe circuite virtuale

Metodele pentru controlul congestiei descrise anterior sunt în principiu în buclă deschisă: ele încearcă, în primul rând, să prevină apariţia congestiei, în loc să o trateze după ce a apărut. În această secţiune, vom descrie unele abordări ale controlului dinamic al congestiei în subreţelele bazate pe circuite virtuale. În următoarele două, vom studia tehnici ce pot fi folosite în orice subreţea.

O tehnică larg răspândită pentru a împiedica agravarea unei congestii deja apărute este **controlul admisiei**. Ideea este simplă: odată ce s-a semnalat apariţia congestiei, nu se mai stabilesc alte circuite virtuale până ce problema nu s-a rezolvat. Astfel, încercarea de a stabili o nouă conexiune la nivel transport eşuează. Lăsând tot mai mulţi utilizatori să stabilească conexiuni, nu ar face decât să agraveze lucrurile. Deşi această abordare este dură, ea este simplă şi uşor de realizat. În sistemul telefonic, dacă o centrală devine supraaglomerată, ea practică controlul admisiei, nemaifurnizând tonul.



O alternativă este de a permite stabilirea de noi circuite virtuale, dar de a dirija cu atenţie aceste noi circuite, ocolind zonele cu probleme. De exemplu, să considerăm subreţeaua din fig. 5-27(a), în care două rutere sunt congestionate, aşa cum se poate observa.

Să presupunem că un calculator gazdă ataşat ruterului *A* doreşte să stabilească o conexiune cu un altul, ataşat ruterului *B*. În mod normal, această conexiune ar trece prin unul dintre cele două rutere congestionate. Pentru a evita această situaţie, putem redesena subreţeaua aşa cum se arată în fig. 5-27(b), omiţând ruterele congestionate şi toate liniile asociate. Linia punctată arată o posibilă cale pentru circuitul virtual, care evită ruterele congestionate.

O altă strategie specifică circuitelor virtuale este negocierea unei înţelegeri între calculatorul gazdă şi subreţea la stabilirea unui circuit virtual. Această înţelegere specifică în mod normal volumul şi forma traficului, calitatea serviciului cerut şi alţi parametri. De obicei, pentru a-şi respecta partea din înţelegere, subreţeaua îşi rezervă resurse de-a lungul căii în momentul stabilirii circuitului virtual. Resursele pot include spaţiu pentru tabele şi zone tampon în rutere şi lăţime de bandă pe linii. În acest fel, congestia nu prea are şanse să se producă pe noul circuit virtual, deoarece toate resursele sunt garantate a fi disponibile.

Acest tip de rezervare poate fi aplicat tot timpul ca o procedură standard sau doar atunci când reţeaua este congestionată. Dezavantajul încercării de a-l face tot timpul este risipa de resurse. Dacă şase circuite virtuale care pot folosi 1 Mbps trec toate prin aceeaşi legătură fizică de 6 Mbps, linia trebuie marcată ca plină, chiar dacă este puţin probabil ca toate cele şase circuite să transmită la nivel maxim în acelaşi timp. În consecinţă, preţul controlului congestiei este lăţime de bandă nefolosită (adică risipită) în cazuri normale.

5.3.4 Controlul congestiei în subretele datagramă

Să ne oprim acum asupra unor abordări care pot fi folosite în subreţelele bazate pe datagrame (dar şi în cele bazate pe circuite virtuale). Fiecare ruter poate gestiona cu uşurinţă folosirea liniilor sale de ieşire precum şi alte resurse. De exemplu, el poate asocia fiecărei linii o variabilă reală, *u,* a cărei valoare, între 0.0 şi 1.0, reflectă utilizarea recentă a acelei linii. Pentru a menţine o estimare cât mai bună a lui *u*, se poate face periodic eşantionarea utilizării instantanee a liniei *f* (fie 0, fie 1) şi apoi actualizarea lui *u* astfel:

$$u = au\ ,, + (1 - a)\ f$$

unde constanta *a* determină cât de repede uită ruterul istoria recentă.

Ori de câte ori *u* depăşeşte pragul, linia de ieşire intră într-o stare de „avertisment”. Fiecare pachet nou-venit este verificat pentru a se vedea dacă linia de ieşire asociată este în starea de avertisment. Dacă este aşa, atunci se iau măsuri. Acţiunea executată poate fi una dintr-o mulţime de alternative, pe care le vom discuta acum.

Bitul de avertizare

Vechea arhitectură DECNET semnala starea de avertizare prin setarea unui bit special în antetul pachetului. La fel procedează şi retransmisia cadrelor (frame relay). Când pachetul ajunge la destinaţia sa, entitatea transport copiază bitul în următoarea confirmare trimisă înapoi la sursă. În acel moment, sursa îşi reduce traficul.

Atât timp cât ruterul a fost în starea de avertizare, acesta a continuat să seteze bitul de avertizare, ceea ce înseamnă că sursa a primit în continuare confirmări cu acest bit setat. Sursa a monitorizat fracţiunea de confirmări cu bitul setat şi şi-a ajustat rata de transmisie corespunzător. Atâta timp cât au continuat să sosească biţi de avertizare, sursa şi-a micşorat continuu rata de transmisie. Odată cu încetinirea sosirii acestora, sursa şi-a mărit rata de transmisie. De observat că, deoarece fiecare ruter de-a lungul căi ar fi putut seta bitul de avertizare, traficul a crescut numai atunci când nici unul dintre rutere nu a avut probleme.

Pachete de soc

Algoritmul anterior pentru controlul congestiei este destul de ingenios. Foloseşte o modalitate ocolită de a înştiinţa sursa să încetinească transmisia. De ce nu-i spune direct? În această abordare, ruterul trimite un **pachet de şoc** către calculatorul gazdă sursă, dându-i destinaţia găsită în pachet. Pachetul original este marcat (un bit din antet este comutat) pentru a nu se mai genera pachete de şoc pe calea aleasă, apoi este retrimis în mod obişnuit.

Un calculator gazdă sursă care primeşte un pachet de şoc trebuie să reducă traficul trimis spre destinaţia specificată cu X procente. Deoarece alte pachete trimise către aceeaşi destinaţie sunt deja pe drum şi vor genera alte pachete de şoc, calculatorul gazdă ar trebui să ignore pachetele de şoc referitoare la destinaţia respectivă o anumită perioadă de timp. După ce perioada s-a scurs, calculatorul gazdă aşteaptă alte pachete de şoc un alt interval. Dacă soseşte un astfel de pachet, linia este încă congestionată, astfel încât calculatorul va reduce fluxul şi mai mult şi va reîncepe să ignore pachetele de şoc. Dacă pe perioada de ascultare nu sosesc pachete de şoc, atunci calculatorul gazdă poate să crească din nou fluxul. Reacţia implicită a acestui protocol poate ajuta la prevenirea congestiei, neştrangulând fluxul decât dacă au apărut probleme.

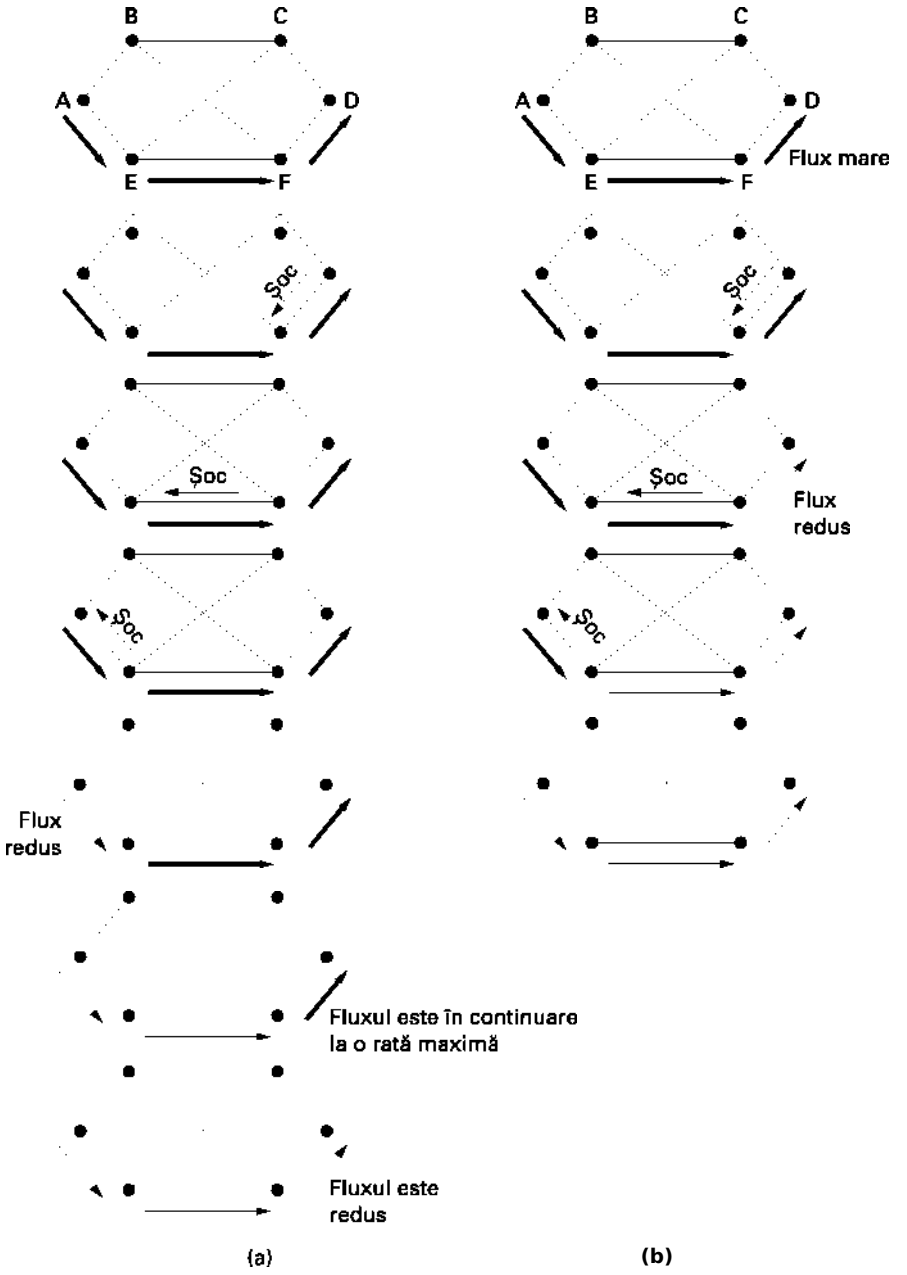


Fig. 5-28. (a) Un pachet de şoc care afectează doar sursa. (b) Un pachet de şoc care afectează fiecare

Calculatorul gazdă poate reduce traficul prin ajustarea parametrilor asociaţi politicii folosite, de exemplu dimensiunea ferestrei. De obicei, primul pachet de şoc

determină scăderea ratei datelor la

0.50 din valoarea anterioară, următoarea reduce traficul la 0.25 şi aşa mai departe. Creşterea valorilor are loc cu rate mai mici pentru a preveni reinstalarea rapidă a congestiei.

Au fost propuse mai multe variante ale acestui algoritm pentru controlul congestiei. În una dintre acestea, fiecare ruter poate menţine mai multe valori de referinţă (praguri). În funcţie de pragul depăşit, pachetul de şoc poate conţine un avertisment blând, unul sever sau un ultimatum.

Altă variantă prevede folosirea lungimilor cozilor sau a utilizării zonelor tampon în locul utilizării liniilor, ca semnal de comutare. Evident, pentru această metrică se poate folosi aceeaşi ponderare exponenţială ca şi pentru *u.*

Pachete de soc salt cu salt

La viteze mari sau pe distanţe mari, trimiterea unui pachet de şoc către calculatorul sursă nu funcţionează normal, reacţia fiind întârziată. Să considerăm, de exemplu, un calculator în San- Francisco (ruterul *A* din fig. 5-28) care trimite trafic către un calculator din New York (ruterul *D* din fig. 5-28) la 155 Mbps. Dacă ruterul din New York rămâne fără zone tampon, atunci pachetului de şoc îi vor trebui circa 30 msec pentru ca să revină la San Francisco solicitând încetinirea. Propagarea pachetului de şoc este prezentată ca al doilea, al treilea şi al patrulea pas din fig. 5-28(a). În aceste 30 msec, se pot trimite 4.6 MB. Chiar atunci când calculatorul gazdă din San Franciso se opreşte imediat, cei 4.6 MB din conductă (eng.: pipe) vor continua să sosească şi trebuie luaţi în seamă. De-abia în a şaptea diagramă din fig. 5-28(a) ruterul din New York va observa reducerea fluxului.

O abordare alternativă este ca pachetele de şoc să aibă efect în fiecare salt prin care trec, aşa cum se arată în secvenţa din fig. 5-28(b). În acest caz, de îndată ce pachetul de şoc ajunge la *F, F* trebuie să reducă fluxul spre *D*. Procedând în acest fel, se cere din partea lui *F* alocarea mai multor zone tampon pentru flux, în timp ce sursa continuă să emită la viteză maximă, însă *D* simte imediat o uşurare, la fel ca într-o reclamă TV pentru înlăturarea durerii de cap. La pasul următor, pachetul de şoc va ajunge la *E,* spunându-i lui *E* să reducă fluxul spre *F*. Această acţiune solicită puternic zonele tampon ale lui *E*, însă are un efect benefic imediat asupra lui *F*, uşurându-i munca. În fine, pachetul de şoc ajunge la *A*, şi fluxul scade efectiv.

Efectul acestei scheme salt cu salt asupra reţelei este asigurarea unei eliberări imediate la locul congestiei cu preţul folosirii mai multor zone tampon. În acest fel congestia poate fi înăbuşită imediat ce se manifestă, fără a se pierde nici un pachet. Ideea este discutată în amănunt în (Mishra şi Kanakia, 1992), unde se furnizează şi rezultatele unei simulări.

5.3.5 Împrăştierea încărcării

Când nici una dintre metodele anterioare nu reduc congestia, ruterele pot să-şi scoată la bătaie artileria grea: împrăştierea încărcării. **Împrăştierea încărcării** este un mod simpatic de a spune că atunci când ruterele sunt inundate de pachete pe care nu le mai pot gestiona, pur şi simplu le aruncă. Termenul provine din domeniul distribuţiei energiei electrice, unde se referă la practica serviciilor publice care lasă intenţionat fără energie electrică anumite zone, pentru a salva întreaga reţea de la

colaps, în zilele călduroase de vară când cererea de energie electrică depăşeşte puternic oferta.

Un ruter care se îneacă cu pachete poate alege la întâmplare pachetele pe care să le arunce, însă de obicei el poate face mai mult decât atât. Alegerea pachetelor care vor fi aruncate poate depinde de aplicaţiile care rulează. Pentru transferul de fişiere, un pachet vechi este mai valoros decât unul nou, deoarece aruncarea pachetului 6 şi păstrarea pachetelor de la 7 la 10 va cauza producerea unei „spărturi" în fereastra receptorului, care poate forţa retrimiterea pachetelor de la 6 la 10 (dacă receptorul distruge automat pachetele care sunt în afara secvenţei). În cazul unui fişier de 12 pachete, aruncarea pachetului 6 poate necesita retransmiterea pachetelor de la 7 la 12, în timp ce aruncarea lui 10 va necesita doar retrimiterea pachetelor de la 10 la 12. În contrast, pentru multimedia, un pachet nou este mult mai important decât un pachet vechi. Prima politică (vechiul este mai bun decât noul) este numită adesea **a vinului**, iar ultima (noul este mai bun decât vechiul) este numită **a laptelui**.

Un pas inteligent mai departe presupune cooperare din partea expeditorilor. Pentru multe aplicaţii, unele pachete sunt mai importante decât altele. De exemplu, unii algoritmi de compresie a imaginilor transmit periodic un cadru întreg şi apoi trimit următoarele cadre ca diferenţe faţă de ultimul cadru complet. În acest caz, aruncarea unui pachet care face parte dintr-o diferenţă este de preferat aruncării unuia care face parte dintr-un cadru întreg. Ca un alt exemplu, să considerăm trimiterea documentelor care conţin text ASCII şi imagini. Pierderea unei linii de pixeli dintr-o imagine este de departe mai puţin dăunătoare decât pierderea unei linii dintr-un text.

Pentru a implementa o politică inteligentă de distrugere a pachetelor, aplicaţiile trebuie să înscrie pe fiecare pachet clasa de prioritate din care face parte, pentru a indica cât de important este. Dacă ele fac aceasta, atunci când trebuie distruse unele pachete, ruterele vor începe cu cele din clasa cea mai slabă, vor continua cu cele din următoarea clasă şi aşa mai departe. Evident, dacă nu ar fi nici un stimulent pentru a marca pachetele şi altfel decât FOARTE IMPORTANT – A NU SE DISTRUGE NICIODATĂ, SUB NICI UN MoTIV, nimeni nu ar face acest lucru.

Stimulentul ar putea fi sub formă de bani, adică pachetele mai puţin prioritare să fie mai ieftin de trimis decât cele cu prioritate mare. Ca alternativă, expeditorilor li s-ar putea permite să trimită pachete cu priorităţi mari cu condiţia ca încărcarea să fie mică, dar odată cu creşterea încărcării acestea ar fi aruncate, încurajând astfel utilizatorii să înceteze trimiterea lor.

O altă opţiune ar fi să se permită calculatoarelor gazdă să depăşească limitele specificate în contractul negociat la iniţializarea circuitului virtual (să folosească o lăţime de bandă mai mare decât li s-a permis), dar cu condiţia ca tot traficul în exces să fie marcat ca fiind de prioritate scăzută. O astfel de strategie nu este de fapt o idee rea, pentru că utilizează în mod mai eficient resursele mai puţin folosite, permiţând astfel calculatoarelor gazdă să le utilizeze atâta timp cât nimeni altcineva nu este interesat, dar fără să stabilească dreptul la ele atunci când situaţia devine mai dură.

Detecţia aleatoare timpurie

Este bine cunoscut faptul că a trata congestia imediat ce a fost detectată este

mai eficient decât să o laşi să-ţi încurce treburile şi apoi să încerci să te descurci cu ea. Această observaţie a dus la ideea de a arunca pachete înainte ca toată zona tampon să fie într-adevăr epuizată. Un algoritm popular pentru a face acest lucru se numeşte **RED** (**Random Early Detection –** rom.: Detecţia aleatoare timpurie) (Floyd şi Jacobson, 1993). În unele protocoale de transport (incluzând TCP-ul), răspunsul la pachetele pierdute este ca sursa să încetinească. Raţionamentul din spatele acestei logici este acela că TCP a fost proiectat pentru reţele cablate, iar acestea sunt foarte sigure, astfel încât pachetele pierdute se datorează mai degrabă depăşirii spaţiului tampon decât erorilor de transmisie. Acest lucru poate fi exploatat pentru a ajuta la reducerea congestiei.

Prin aruncarea pachetelor de către rutere înainte ca situaţia să devină fără speranţă (de unde şi termenul "timpuriu" din denumire), ideea este de a lua măsuri înainte de a fi prea târziu. Pentru a determina când să înceapă aruncarea, ruterele menţin neîntrerupt o medie a lungimilor cozilor lor. Când lungimea medie a cozii unei linii depăşeşte o limită, se spune că linia este congestionată şi se iau măsuri.

Având în vedere că ruterul probabil că nu poate spune care sursă produce necazul cel mai mare, alegerea la întâmplare a unui pachet din coada care a determinat această acţiune este probabil cel mai bun lucru pe care îl poate face.

Cum ar trebui ruterul să informeze sursa despre problemă? O cale este aceea de a-i trimite un pachet de şoc, după cum am descris. O problemă a acestei abordări este că încarcă şi mai mult reţeaua deja congestionată. O strategie diferită este să arunce pur şi simplu pachetul selectat şi să nu raporteze acest lucru. Sursa va observa în cele din urmă lipsa confirmării şi va lua măsuri. Deoarece ştie că pachetele pierdute sunt în general determinate de congestie şi aruncări, va reacţiona prin încetinire în loc să încerce mai abitir. Această formă implicită de reacţie funcţionează doar atunci când sursele răspund la pachetele pierdute prin reducerea ratei lor de transmisie. În reţele fără fir, unde majoritatea pierderilor se datorează zgomotelor legăturii aeriene, această abordare nu poate fi folosită.

5.3.6 Controlul fluctuaţiilor

Pentru aplicaţii de genul transmisiilor audio sau video, nu prea contează dacă pachetele au nevoie de 20 msec sau de 30 msec pentru a fi distribuite, atât timp cât durata de tranzit este constantă. Variaţia (adică deviaţia standard) timpului de sosire a pachetului se numeşte **fluctuaţie.** O fluctuaţie mare, de exemplu dacă unele pachete au nevoie de 20 msec şi altele de 30 msec pentru a ajunge, va produce sunet sau imagine de calitate inegală. Fluctuaţia este ilustrată în fig. 5-29. În contrast, o înţelegere ca 99 procente din pachete să fie livrate cu o întârziere de la 24.5 msec până la 25.5 msec ar putea fi acceptabilă.

Limita aleasă trebuie să fie, bineînţeles, reală. Ea trebuie să ia în calcul durata de tranzit a vitezei luminii şi întârzierea minimă prin rutere şi poate să-şi lase o mică rezervă pentru unele întârzieri inevitabile.

Fluctuaţiile pot fi limitate prin calcularea timpului de tranzit estimat pentru fiecare salt de-a lungul căii. Când un pachet ajunge la ruter, ruterul verifică să vadă cât de mult este decalat pachetul faţă de planificarea sa. Această informaţie este păstrată în pachet şi actualizată la fiecare salt. Dacă pachetul a sosit înaintea

planificării, el este ţinut suficient pentru a reveni în grafic. Dacă pachetul este întârziat faţă de planificare, ruterul încearcă să-l trimită cât mai repede.

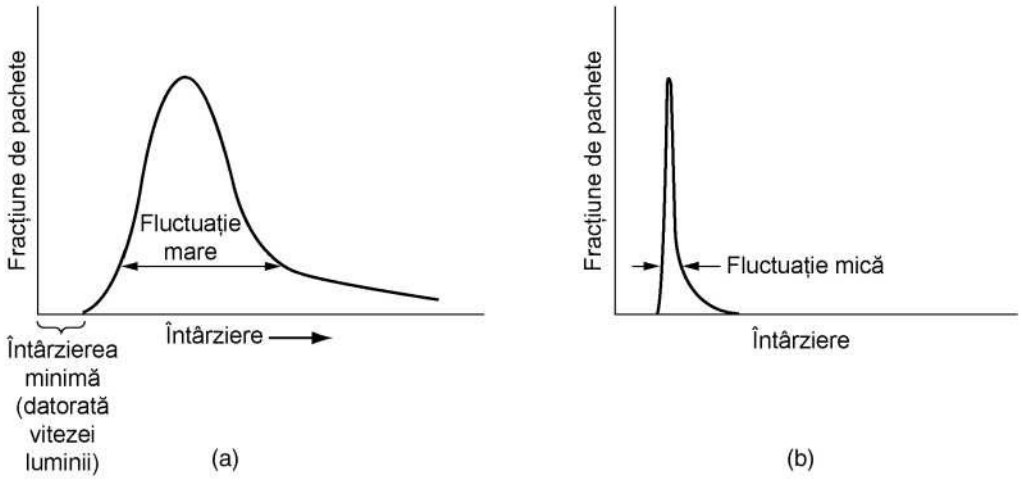


Fig. 5-29. (a) Fluctuaţie mare. (b) Fluctuaţie scăzută.

De fapt, algoritmul pentru determinarea pachetului care va merge mai departe dintr-un set de pachete care concurează pentru o linie de ieşire, va alege întotdeauna pachetul cu cea mai mare întârziere faţă de program. În acest fel, pachetele care au ajuns mai repede sunt încetinite, iar cele care sunt întârziate vor fi accelerate, în ambele cazuri reducându-se fluctuaţia.

În unele aplicaţii, cum ar fi aplicaţii video la cerere, fluctuaţiile pot fi eliminate prin adăugarea unei zone tampon la receptor şi apoi aducerea datelor pentru a fi afişate din zona tampon în loc de a fi luate din reţea în timp real. Totuşi, pentru alte aplicaţii, în special acelea care necesită interacţiune în timp real între oameni, cum ar fi convorbirile telefonice şi videoconferinţele prin Internet, întârzierea inerentă utilizării zonelor tampon este inacceptabilă. Controlul congestiei este o arie activă a cercetării. Nivelul actual este rezumat în (Gevros et al., 2001).

5.4 CALITATEA SERVICIILOR

Tehnicile pe care le-am analizat în secţiunile precedente sunt proiectate pentru a reduce congestia şi a îmbunătăţi performanţele reţelei. Cu toate acestea, odată cu creşterea lucrului cu aplicaţii multimedia în reţea, de multe ori aceste măsuri luate ad-hoc nu sunt de ajuns. Este nevoie de preocupări serioase în proiectarea reţelelor şi a protocoalelor pentru garantarea calităţii serviciilor. În secţiunile următoare ne vom continua studiul asupra performanţei reţelei, dar acum ne vom concentra mai mult asupra modalităţilor de a furniza o calitate a serviciilor în conformitate cu necesităţile fiecărei aplicaţii. Ar trebui stipulat de la început faptul că, totuşi, multe dintre aceste idei sunt în formare şi sunt supuse schimbărilor.

5.4.1 Cerinţe

Un şir de pachete de la o sursă la o destinaţie se numeşte **flux**. Într-o reţea orientată pe conexiune, toate pachetele aparţinând unui flux merg pe aceeaşi rută; într-o reţea neorientată pe conexiune, acestea pot urma rute diferite. Necesităţile fiecărui flux pot fi caracterizate prin patru parametri primari: fiabilitatea, întârzierea, fluctuaţia şi lăţimea de bandă. Împreună, acestea determină **QoS** (**Quality of Service –** rom.: Calitatea serviciilor) pe care o necesită fluxul. În fig. 5-30 sunt listate câteva aplicaţii obişnuite precum şi cerinţele stringente ale acestora.

Aplicaţia	Fiabilitatea	Întârzierea	Fluctuaţia	Lăţimea de
Poşta electronică	Mare	Mică	Mică	Mică
Transfer de fişiere	Mare	Mică	Mică	Medie
Acces Web	Mare	Medie	Mică	Medie
Conectare la distanţă	Mare	Medie	Medie	Mică
Audio la cerere	Mică	Mică	Mare	Medie
Video la cerere	Mică	Mică	Mare	Mare
Telefonie	Mică	Mare	Mare	Mică
Videoconferinţe	Mică	Mare	Mare	Mare

Fig. 5-30. Cât de stringente sunt cerinţele referitoare la calitatea serviciilor

Primele patru aplicaţii au cerinţe stringente de fiabilitate. Nu se accepta livrarea de biţi incorecţi. Acest obiectiv este atins de obicei prin crearea unei sume de control pentru fiecare pachet şi verifica
rea acestei sume de control la destinaţie. Dacă un pachet este alterat pe parcurs, nu este confirmat şi în cele din urmă va fi retransmis. Această strategie oferă o fiabilitate mare. Ultimele patru aplicaţii (audio/video) pot tolera erori, aşa că nu este calculată sau verificată nici o sumă de control.

Aplicaţiile de transfer de fişiere, inclusiv poşta electronică şi video nu sunt sensibile la întârzieri. Dacă toate pachetele sunt întârziate uniform cu câteva secunde, nu este nici o problemă. Aplicaţiile interactive, cum ar fi navigarea pe Web şi conectarea la distanţă, sunt mult mai sensibile la întârzieri. Aplicaţiile de timp real, cum ar fi telefonia şi videoconferinţele au cerinţe stricte de întârziere. Dacă toate cuvintele dintr-o convorbire telefonică sunt fiecare întârziate cu 2.000 secunde, utilizatorii vor găsi conexiunea inacceptabilă. Pe de altă parte, rularea de fişiere audio sau video de pe server nu cere întârzieri scăzute.

Primele trei aplicaţii nu sunt sensibile la sosirea pachetelor la intervale neregulate. Conectarea la distanţă este într-un fel sensibilă la aceasta deoarece caracterele vor apărea pe ecran în rafale dacă conexiunea are multe fluctuaţii. Aplicaţiile video şi în special cele audio sunt extrem de sensibile la fluctuaţii. Dacă un utilizator urmăreşte un fişier video din reţea şi cadrele sunt toate întârziate cu exact 2.000 secunde, nu este nici o problemă. Dar dacă timpul de transmisie variază aleator între 1 şi 2 secunde, rezultatul va fi îngrozitor. Pentru audio, o fluctuaţie chiar şi de câteva milisecunde este perfect sesizabilă.

În cele din urmă, aplicaţiile diferă la cerinţele de lăţime de bandă, fără ca poşta electronică şi conectarea la distanţă să necesite o bandă prea largă, iar aplicaţiile video de orice fel necesitând o bandă foarte largă. Reţelele ATM clasifică fluxul în patru mari categorii, în funcţie de cerinţele lor de QoS, după cum urmează:

1. Rată de transfer constantă (de exemplu telefonie).
2. Rată de transfer variabilă în timp real (de exemplu videoconferenţiere compresată).
3. Rată de transfer variabilă, dar nu în timp real (de exemplu vizionarea unui film pe Internet).
4. Rată de transfer disponibilă (de exemplu transfer de fişiere).

Aceste categorii sunt de asemenea folositoare pentru alte scopuri şi pentru alte tipuri de reţele. Rata de transfer constantă este o încercare de a simula un cablu, furnizând o lăţime de bandă uniformă şi o întârziere uniformă. Rata de transfer variabilă apare pentru video compresat, unele cadre fiind mai comprimate decât altele. Astfel, transmisia unui cadru cu o mulţime de detalii în el ar putea necesita transmisia mai multor biţi, în timp ce transmisia imaginii unui perete alb s-ar putea comprima extrem de bine. Rata de transfer disponibilă este pentru aplicaţii care nu sunt sensibile la întârzieri sau la fluctuaţii, cum ar fi poşta electronică.

5.4.2 Tehnici pentru obţinerea unei bune calităţi a serviciilor

Acum că ştim câte ceva despre cerinţele QoS, cum le obţinem? Ei bine, pentru început, nu există o reţetă magică. Nici o tehnică nu furnizează într-un mod optim o calitate a serviciilor eficientă şi pe care să te poţi baza. În schimb, au fost dezvoltate o varietate de tehnici, cu soluţii practice care adeseori combină mai multe tehnici. Vom examina acum câteva dintre tehnicile folosite de proiectanţii de sisteme pentru obţinerea calităţii serviciilor.

Supraaprovizionarea

O soluţie uşoară este aceea de a furniza ruterului suficientă capacitate, spaţiu tampon şi lăţime de bandă încât pachetele să zboare pur şi simplu. Problema cu această soluţie este aceea că este costisi

toare. Odată cu trecerea timpului şi cu faptul că proiectanţii au o idee mai clară despre cât de mult înseamnă suficient, această tehnică ar putea deveni chiar realistă. Până la un punct, sistemul telefonic este supraapovizionat Se întâmplă rar să ridici receptorul telefonului şi să nu ai ton de formare instantaneu. Pur şi simplu există atâta capacitate disponibil ă încât cererea va fi întotdeauna satisfăcută.

Memorarea temporară

Fluxurile pot fi reţinute în zone tampon ale receptorului înainte de a fi livrate. Aceasta nu afectează fiabilitatea sau lăţimea de bandă şi măreşte întârzierea, dar uniformizează fluctuaţia. Pentru audio şi video la cerere, fluctuaţiile reprezintă problema principală, iar această tehnică este de mare ajutor.

Am văzut diferenţa dintre fluctuaţiile mari şi fluctuaţiile mici în fig. 5-29. În fig. 5-31 vedem un flux de pachete care sunt livrate cu o fluctuaţie considerabilă. Pachetul 1 este transmis de către server la momentul *t = 0* sec şi ajunge la client la *t = 1* sec. Pachetul 2 acumulează o întârziere mai mare şi îi sunt necesare 2 sec pentru a ajunge. Pe măsură ce pachetele sosesc, ele sunt reţinute în zona tampon pe maşina client.

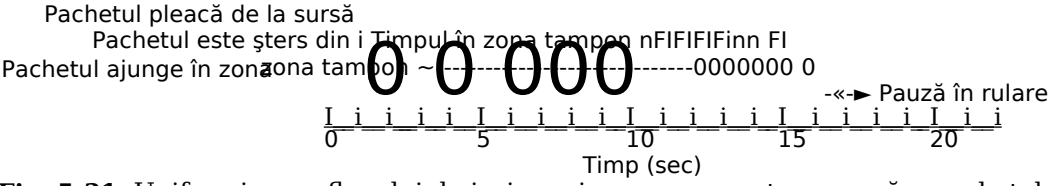


Fig. 5-31. Uniformizarea fluxului de ieşire prin memorarea temporară a pachetelor.

La *t = 10* sec începe rularea. La acest moment, pachetele de la 1 la 6 au fost introduse în zona tampon aşa că pot fi scoase de acolo la intervale egale pentru o rulare uniformă. Din păcate, pachetul 8 a avut o întârziere aşa de mare încât nu este disponibil când îi vine timpul de rulare, aşa că rularea trebuie întreruptă până la sosirea acestuia, creând astfel o pauză supărătoare în muzică sau în film. Această problemă poate fi alinată prin întârzierea şi mai mare a momentului începerii, deşi acest lucru necesită şi un spaţiu tampon mai mare. Sit-urile Web comerciale care conţin fluxuri audio sau video folosesc toate programe de rulare care au timpul de reţinere în zona tampon de aproximativ 10 secunde până să înceapă să ruleze aplicaţia.

Formarea traficului

În exemplul de mai sus, sursa trimite pachetele la intervale de timp egale între ele, dar în alte cazuri ele pot fi emise neregulat, ceea ce poate duce la apariţia congestiei în reţea. Neuniformitatea ieşirii este un lucru obişnuit dacă server-ul se ocupă de mai multe fluxuri la un moment dat şi de asemeni permite şi alte acţiuni cum ar fi derularea rapidă înainte sau înapoi, autentificarea utilizatorilor şi aşa mai departe. De asemenea, abordarea pe care am folosit-o aici (memorarea temporară) nu este întotdeauna posibilă, de exemplu în cazul videoconferinţelor. Cu toate acestea, dacă s-ar putea face ceva pentru ca serverul (şi calculatoarele gazdă în general) să transmită cu rate de transfer uniforme, calitatea serviciilor ar fi mai bună. Vom examina acum o tehnică numită **formarea traficului (traffic shaping)**, care uniformizează traficul mai degrabă pentru server decât pentru client.

Formarea traficului se ocupă cu uniformizarea ratei medii de transmisie a datelor (atenuarea rafalelor). În contrast, protocoalele cu fereastră glisantă pe care le-am studiat anterior limitează volumul de date în tranzit la un moment dat şi nu rata la care sunt transmise acestea. La momentul stabilirii unei conexiuni, utilizatorul şi subreţeaua (clientul şi furnizorul) stabilesc un anumit model al traficului (formă) pentru acel circuit. În unele cazuri aceasta se numeşte **înţelegere la nivelul serviciilor (service level agreement)**. Atât timp cât clientul îşi respectă partea sa de contract şi trimite pachete conform înţelegerii încheiate, furnizorul promite livrarea lor în timp util. Formarea traficului reduce congestia şi ajută furnizorul să-şi ţină promisiunea. Astfel de înţelegeri nu sunt foarte importante pentru transferul de fişiere, însă sunt deosebit de importante pentru datele în timp real, cum ar fi conexiunile audio sau video, care au cerinţe stringente de calitate a serviciilor.

Pentru formarea traficului clientul spune furnizorului: „Modelul meu de transmisie arată cam aşa. Poţi să te descurci cu el?” Dacă furnizorul este de acord, problema care apare este cum poate spune furnizorul dacă clientul respectă înţelegerea şi ce să facă dacă nu o respectă. Monitorizarea fluxului traficului se numeşte **supravegherea traficului (traffic policing.)** Stabilirea unei forme a traficului şi urmărirea respectării ei se fac mai uşor în cazul subreţelelor bazate pe circuite virtuale decât în cazul subreţelelor bazate pe datagrame. Cu toate acestea, chiar şi în cazul subreţelelor bazate pe datagrame, aceleaşi idei pot fi aplicate la conexiunile nivelului transport.

Algoritmul găleţii găurite

Să ne imaginăm o găleată cu un mic orificiu în fundul său, aşa cum este prezentată în fig. 5-32(a). Nu contează cu ce rată curge apa în găleată, fluxul de ieşire va fi la o rată constantă, p, dacă există apă în găleată şi zero dacă găleata este goală. De asemenea, odată ce găleata s-a umplut, orice cantitate suplimentară de apă se va revărsa în afara pereţilor şi va fi pierdută (adică nu se va regăsi în fluxul de ieşire de sub orificiu).

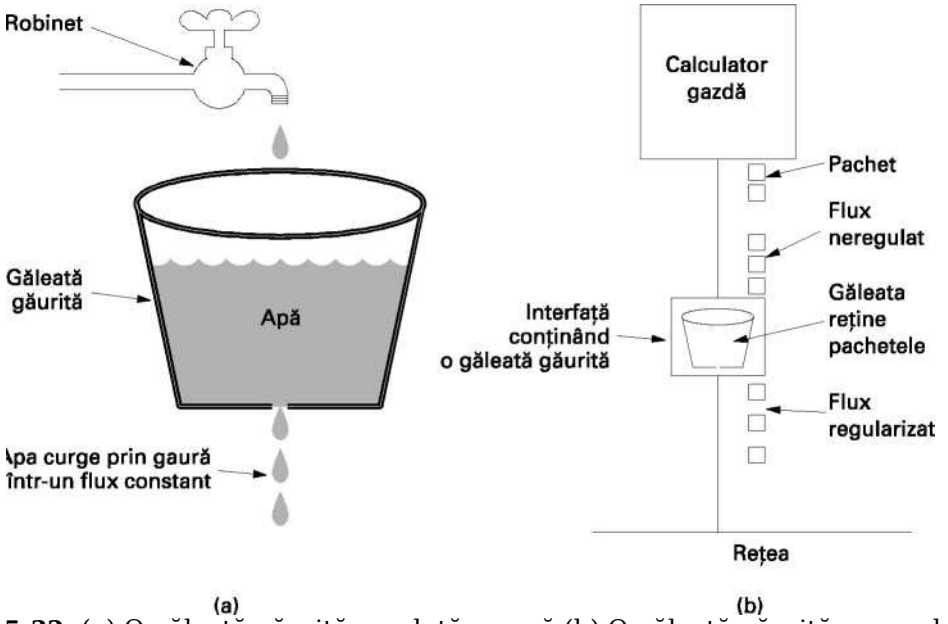


Fig. 5-32. (a) O găleată găurită umplută cu apă (b) O găleată găurită, cu pachete.

Aceeaşi idee poate fi aplicată şi în cazul pachetelor, aşa cum se arată în fig. 5-32 (b). Conceptual, fiecare calculator gazdă este conectat la reţea printr-o interfaţă conţinând o găleată găurită, în fapt o coadă internă cu capacitate finită. Dacă un pachet soseşte în coadă atunci când aceasta este plină, el este distrus. Cu alte cuvinte, dacă unul sau mai multe procese de pe calculatorul gazdă încearcă trimiterea unui pachet atunci când coada conţine deja numărul maxim de pachete, pachetele noi vor fi distruse fără menajamente. Acest aranjament poate fi implementat în interfaţa hardware sau poate fi simulat de către sistemul de operare gazdă. A fost propus pentru prima dată de către Turner (1986) şi este numit **algoritmul găleţii găurite (the leaky bucket algorithm)**. De fapt nu este altceva decât un sistem de cozi cu un singur server şi cu timp de servire constant.

Calculatorul gazdă poate pune în reţea câte un pachet la fiecare tact al ceasului. Şi acest lucru poate fi realizat de placa de interfaţă sau de către sistemul de operare. Acest mecanism transformă un flux neregulat de pachete de la procesele de pe calculatorul gazdă într-un flux uniform de pachete care se depun pe reţea, netezind rafalele şi reducând mult şansele de producere a congestiei.

Dacă toate pachetele au aceeaşi dimensiune (de exemplu celule ATM), algoritmul poate fi folosit exact aşa cum a fost descris. Dacă se folosesc pachete de lungimi variabile, este adesea mai convenabil să se transmită un anumit număr de octeţi la fiecare tact şi nu un singur pachet. Astfel, dacă regula este 1024 octeţi la fiecare tact, atunci se pot transmite un pachet de 1024 octeţi, două de 512 octeţi, sau patru de 256 octeţi ş.a.m.d. Dacă numărul rezidual de octeţi este prea mic, următorul pachet va trebui să aştepte următorul tact.

Implementarea algoritmului iniţial al găleţii găurite este simplă. Găleata găurită constă de fapt dintr-o coadă finită. Dacă la sosirea unui pachet este loc în coadă, el este adăugat la sfârşitul cozii, în caz contrar, este distrus. La fiecare tact se trimite un pachet din coadă (bineînţeles dacă aceasta nu este vidă).

Algoritmul găleţii folosind contorizarea octeţilor este implementat aproximativ în aceeaşi manieră. La fiecare tact un contor este iniţializat la *n.* Dacă primul pachet din coadă are mai puţini octeţi decât valoarea curentă a contorului, el este transmis şi contorul este decrementat cu numărul corespunzător de octeţi. Mai pot fi transmise şi alte pachete adiţionale, atât timp cât contorul este suficient de mare. Dacă contorul scade sub lungimea primului pachet din coadă, atunci transmisia încetează până la următorul tact, când contorul este reiniţializat şi fluxul poate continua.

Ca un exemplu de găleată găurită, să ne imaginăm un calculator care produce date cu rata de 25 milioane octeţi/sec (200 Mbps) şi o reţea care funcţionează la aceeaşi viteză. Cu toate acestea, ruterele pot să gestioneze datele la această rată doar pentru un timp foarte scurt (practic, până când li se umple spaţiul tampon). Pentru intervale mai mari ele lucrează optim pentru rate care nu depăşesc valoarea de 2 milioane octeţi/sec. Să presupunem acum că datele vin în rafale de câte 1 milion de octeţi, câte o rafală de 40 msec în fiecare secundă. Pentru a reduce rata medie la 2 MB/sec, putem folosi o găleată găurită având p = 2 MB/sec şi o capacitate, *C,* de 1 MB. Aceasta înseamnă că rafalele de până la 1 MB pot fi gestionate fără pierderi de date şi că acele rafale sunt împrăştiate de-a lungul a 500 msec, indiferent cât de repede sosesc.

În fig. 5-33(a) vedem intrarea pentru găleata găurită funcţionând la 25 MB/sec pentru 40 msec. În fig. 5-33(b) vedem ieşirea curgând la o rată uniformă de 2 MB/sec pentru 500 msec.

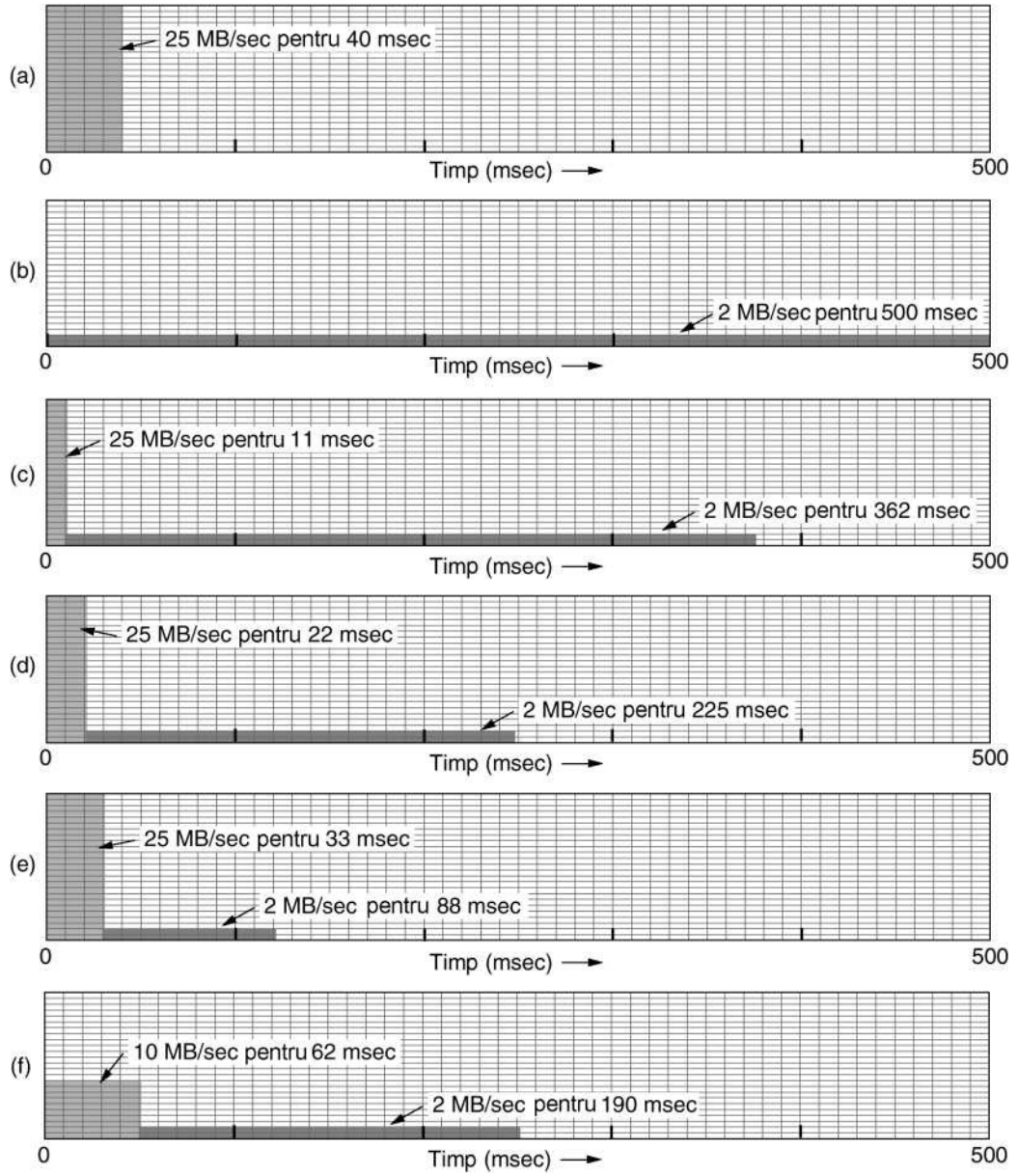


Fig. 5-33. (a) Intrarea pentru o găleată găurită. (b) Ieşirea pentru o găleată găurită. (c) – (e) Ieşirea pentru o găleată cu jeton de capacitate 250 KB, 500 KB, 750 KB. (f) Ieşirea pentru o găleată cu jeton de capacitate 500 KB care alimentează o găleată găurită de 10 MB/sec.

Algoritmul găleţii cu jeton

Algoritmul găleţii găurite impune un model rigid al ieşirii, din punct de vedere al ratei medii, indiferent de cum arată traficul. Pentru numeroase aplicaţii este mai convenabil să se permită o creştere a vitezei de ieşire la apariţia unor rafale mari, astfel încât este necesar un algoritm mai flexibil, de preferat unul care nu pierde date. Un astfel de algoritm este **algoritmul găleţii cu jeton (the token bucket**

algorithm). În acest algoritm, găleata găurită păstrează jetoane, generate de un ceas cu rata de un jeton la fiecare AT sec. În fig. 5-34(a) vedem o găleată păstrând trei jetoane şi având cinci pachete care aşteaptă să fie transmise. Pentru ca un pachet să fie transmis, el trebuie să captureze şi să distrugă un jeton. În fig. 5-34(b) vedem că trei din cele cinci pachete au trecut mai departe, în timp ce celelalte două aşteaptă să fie generate alte două jetoane.

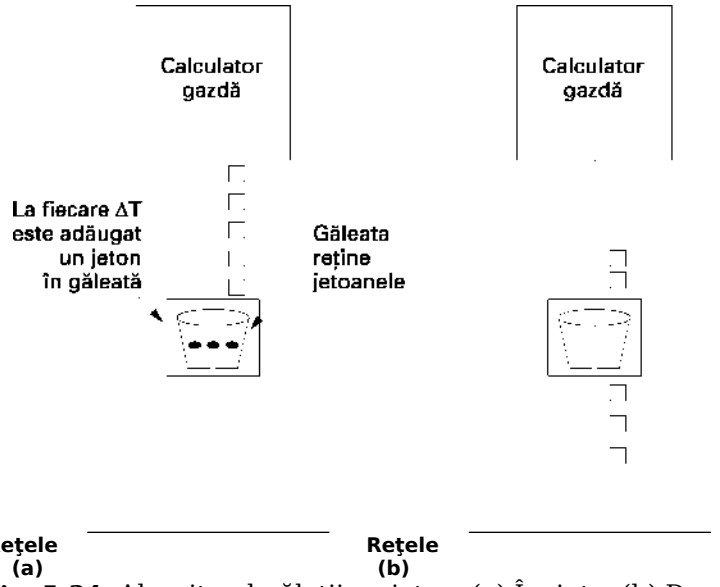


Fig. 5-34. Algoritmul găleţii cu jeton. (a) Înainte. (b) După.

Algoritmul găleţii cu jeton asigură o formare diferită a traficului, comparativ cu algoritmul găleţii găurite. Algoritmul găleţii găurite nu permite calculatoarelor gazdă inactive să acumuleze permisiunea de a trimite rafale mari ulterior. Algoritmul găleţii cu jeton permite această acumulare, mergând până la dimensiunea maximă a găleţii, *n*. Această proprietate permite ca rafale de până la *n* pachete să fie trimise dintr-o dată, permiţând apariţia unor rafale la ieşire şi asigurând un răspuns mai rapid la apariţia bruscă a unor rafale la intrare.

O altă diferenţă între cei doi algoritmi este aceea că algoritmul găleţii cu jeton aruncă jetoanele (capacitatea de transmisie) la umplerea găleţii, dar niciodată nu distruge pachete. Prin contrast, algoritmul găleţii găurite distruge pachete la umplerea găleţii.

Şi aici este posibilă o variantă, în care fiecare jeton reprezintă dreptul de a trimite nu un pachet, ci *k* octeţi. Un pachet va putea fi trimis doar dacă se deţin suficiente jetoane pentru a-i acoperi lungimea exprimată în octeţi. Jetoanele fracţionare sunt păstrate pentru utilizări ulterioare.

Algoritmul găleţii găurite şi algoritmul găleţii cu jeton pot fi folosite şi pentru a uniformiza traficul între rutere, la fel de bine cum pot fi folosite pentru a regla informaţia de ieşire a unui calculator gazdă, ca în exemplul prezentat. Oricum, o diferenţă evidentă este aceea că algoritmul găleţii cu jeton folosit pentru a regla ieşirea unui calculator gazdă îl poate opri pe acesta să trimită date atunci când apare vreo restricţie. A spune unui ruter să oprească transmisiile în timp ce datele

de intrare continuă să sosească, poate duce la pierderea de date.

Implementarea de bază a algoritmului găleţii cu jeton se face printr-o variabilă care numără jetoane. Acest contor creşte cu 1 la fiecare AT şi scade cu 1 ori de câte ori este trimis un pachet. Dacă acest contor atinge valoarea zero, nu mai poate fi trimis nici un pachet. În varianta la nivel de octeţi, contorul este incrementat cu k octeţi la fiecare AT şi decrementat cu lungimea pachetului trimis.

Caracteristica esenţială a algoritmului găleţii cu jeton este că permite rafalele, dar până la o lungime maximă controlată. Priviţi de exemplu fig. 5-33(c). Aici avem o găleată cu jeton, de capacitate 250 KB. Jetoanele sosesc cu o rată care asigură o ieşire de 2 MB/sec. Presupunând că găleata este plină când apare o rafală de 1 MB, ea poate asigura scurgerea la întreaga capacitate de 25 MB/sec pentru circa 11 milisecunde. Apoi trebuie să revină la 2 MB/sec până ce este trimisă întreaga rafală.

Calculul lungimii rafalei de viteză maximă este puţin mai special. Nu este doar 1 MB împărţit la 25 MB/sec deoarece, în timp ce rafala este prelucrată, apar alte jetoane. Dacă notăm S cu lungimea rafalei în secunde, cu C capacitatea găleţii în octeţi, cu p rata de sosire a jetoanelor în octeţi/secundă, cu M rata maximă de ieşire în octeţi/secundă, vom vedea că o rafală de ieşire conţine cel mult $C + \mathrm{p}S$ octeţi. De asemenea, mai ştim că numărul de octeţi într-o rafală de viteză maximă de S secunde este MS. De aici avem:

$$C + \mathrm{p}S = MS$$

Rezolvând această ecuaţie obţinem $S = C / (M - \mathrm{p})$. Pentru parametrii consideraţi C = 250 KB, M = 25 MB/sec şi p = 2 MB/sec vom obţine o durată a rafalei de circa 11 msec. Fig. 5-33(d) şi fig. 5- 33(e) prezintă găleţile cu jeton de capacităţi 500 KB şi respectiv 750 KB.

O problemă potenţială a algoritmului găleţii cu jeton este aceea că şi el permite apariţia unor rafale mari, chiar dacă durata maximă a unei rafale poate fi reglată prin selectarea atentă a lui p şi M. De multe ori este de dorit să se reducă valoarea de vârf, dar fără a se reveni la valorile scăzute permise de algoritmul original, al găleţii găurite.

O altă cale de a obţine un trafic mai uniform este de a pune o găleată găurită după cea cu jeton. Rata găleţii găurite va trebui să fie mai mare decât parametrul p al găleţii cu jeton, însă mai mică decât rata maximă a reţelei. Fig. 5-25(f) ilustrează comportarea unei găleţi cu jeton de 500 KB, urmată de o găleată găurită de 10 MB/sec.

Gestionarea tuturor acestor scheme poate fi puţin mai specială. În esenţă, reţeaua trebuie să simuleze algoritmul şi să se asigure că nu se trimit mai multe pachete sau mai mulţi octeţi decât este permis. Totuşi, aceste instrumente furnizează posibilităţi de a aduce traficul reţelei la forme mai uşor de administrat pentru a ajuta la îndeplinirea condiţiilor necesare calităţii serviciilor.

Rezervarea resurselor

Un bun început pentru garantarea calităţii serviciilor este capabilitatea de a regla forma traficului oferit. Totuşi, folosirea efectivă a acestei informaţii înseamnă implicit să ceri ca toate pachetele dintr-un flux să urmeze aceeaşi rută. Răspândirea lor aleatoare pe rutere face dificil de garantat orice. Ca o consecinţă, trebuie ca

între sursă şi destinaţie să se creeze ceva similar unui circuit virtual şi ca toate pachetele care aparţin fluxului să urmeze această cale.

Odată ce avem o rută specifică pentru un flux, devine posibil să rezervi resurse de-a lungul acelei rute pentru a te asigura că este disponibilă capacitatea necesară. Pot fi rezervate trei tipuri diferite de resurse:

1. Lăţimea de bandă
2. Zona tampon
3. Ciclurile procesorului

Prima, lăţimea de bandă este cea mai evidentă. Dacă un flux cere 1 Mbps şi linia pe care se iese are capacitatea de 2 Mbps, încercarea de a dirija trei fluxuri prin acea linie nu va reuşi. Astfel, a rezerva lăţime de bandă înseamnă să nu se supraîncarce vreo linie de ieşire.

O a doua resursă care este adeseori insuficientă este spaţiul tampon. Când soseşte un pachet, acesta este de obicei depozitat pe placa de reţea de către hardware. Software-ul ruterului trebuie să copieze apoi pachetul într-o zonă tampon din RAM şi să adauge acest tampon în coada pentru transmisie pe linia de ieşire aleasă. Dacă nu este disponibil nici un tampon, pachetul trebuie aruncat deoarece nu există spaţiu în care să poată fi pus. Pentru o bună calitate a serviciilor, unele zone tampon pot fi rezervate pentru un anumit flux în aşa fel încât acel flux să nu trebuiască să concureze pentru tampoane cu alte fluxuri. Întotdeauna va exista o zonă tampon disponibilă atunci când fluxul va avea nevoie, dar până la o anumită limită.

În cele din urmă, ciclurile procesorului sunt de asemenea resurse rare. Ruterul are nevoie de timp de procesor pentru a prelucra un pachet, deci un ruter poate procesa doar un anumit număr de pachete pe secundă. Este necesar să ne asigurăm că procesorul nu este supraîncărcat pentru a asigura prelucrarea în timp util a fiecărui pachet.

La prima vedere ar putea părea că dacă, să zicem, un ruter are nevoie de 1 psec pentru a procesa un pachet, atunci el poate procesa 1 milion de pachete/sec. Această observaţie nu este adevărată pentru că vor exista întotdeauna perioade nefolosite datorate fluctuaţiilor statistice ale încărcării. Dacă procesorul are nevoie de fiecare ciclu de ceas pentru a-şi face treaba, pierderea chiar şi a câtorva cicluri din cauza perioadelor de nefolosire ocazionale creează un decalaj de care nu se poate scăpa.

Oricum, chiar şi cu o încărcare puţin sub capacitatea teoretică, se pot forma cozi şi pot apărea întârzieri. Să considerăm o situaţie în care pachetele sosesc aleator, cu rată medie de sosire de X pachete/sec. Timpul de procesor cerut de fiecare pachet este de asemenea aleator, cu o capacitate medie de procesare de p pachete/sec. Presupunând că distribuţiile sosirii şi servirii sunt distribuţii Poisson, folosind teoria cozilor se poate demonstra că întârzierea medie a unui pachet, *T,* este:

$$= \frac{11}{M^{1} - P} x\text{-----}$$

unde p=X/p este utilizarea procesorului. Primul factor, 1/p, reprezintă timpul de servire în absenţa competiţiei. Al doilea factor este încetinirea cauzată de competiţia cu alte fluxuri. De exemplu, dacă X = 950,000 de pachete/sec şi p = 1,000,000 de pachete/sec, atunci p = 0.95 şi întârzierea medie a unui pachet va fi de 20 psec în loc de 1 psec Acest timp ia în considerare atât timpul de aşteptare în coadă, cât şi timpul de servire, după cum se poate vedea atunci când încărcarea este foarte mică (X/p ® 0). Dacă presupunem că pe traseul fluxului există 30 de rutere, numai datorită întârzierilor în cozi vor rezulta 600psec de întârziere.

Controlul accesului

Acum suntem în punctul în care traficul dintr-un flux oarecare este bine format şi poate să urmeze o singură rută în care capacitatea poate fi rezervată în avans pe ruterele ce se găsesc de-a lungul căii. Când un asemenea flux este oferit unui ruter, acesta trebuie să decidă, pe baza capacităţii sale şi a numărului de angajamente pe care le-a făcut deja cu alte fluxuri, dacă să primească sau să respingă fluxul.

Decizia de a accepta sau de a respinge fluxul nu este o simplă chestiune de comparaţie între (lăţime de bandă, zone tampon, cicluri) cerute de către flux şi capacitatea în exces a ruterului în aceste trei dimensiuni. Este puţin mai complicat decât atât. În primul rând, deşi unele aplicaţii şi-ar putea cunoaşte necesităţile de lăţime de bandă, puţine ştiu despre zonele tampon sau despre ciclurile de procesor, deci, la nivel minim, este necesar un alt mod de descriere a fluxurilor. Apoi, unele aplicaţii sunt de departe mai tolerante la ratarea ocazională a unui termen limită. În cele din urmă, unele aplicaţii ar putea fi dornice să negocieze în legătură cu parametrii fluxului, iar altele nu. De exemplu, un server video care rulează în mod normal la 30 de cadre/sec ar putea fi dispus să coboare la 25 de cadre /sec dacă nu există suficientă lăţime de bandă liberă pentru a suporta 30 de cadre/sec. În mod similar pot fi ajustaţi numărul de pixeli pe cadru, lăţimea de bandă audio şi alte proprietăţi.

Deoarece în negocierea fluxului pot fi implicate mai multe părţi (emiţătorul, receptorul şi toate ruterele aflate de-a lungul căii dintre aceştia), fluxurile trebuie descrise exact în termenii parametrilor specifici ce pot fi negociaţi. Un set de astfel de parametri se numeşte **specificarea fluxului (flow specification)**. Tipic, emiţătorul (adică serverul video) produce o specificaţie a fluxului propunând parametrii pe care ar vrea să îi folosească. Pe măsură ce specificaţia se propagă de-a lungul căii, fiecare ruter o examinează şi îi modifică parametrii după cum are nevoie. Modificările pot doar să reducă fluxul, nu să îl şi crească (de exemplu o rată mai mică a datelor, nu mai mare). Când ajunge la capătul celălalt, parametrii pot fi stabiliţi.

Ca un exemplu de ce poate exista într-o specificaţie a fluxului, să considerăm exemplul din fig. 535, care se bazează pe RFC-urile 2210 şi 2211. Are cinci parametri, dintre care primul, *rata găleţii cu jeton,* reprezintă numărul de octeţi pe secundă care sunt puşi în găleată. Aceasta este rata maximă suportată la care poate transmite emiţătorul, mediată de-a lungul unui interval mare de timp.

Parametru	Unitate de măsură
Rata găleţii cu jeton	Octeţi/sec
Dimensiunea găleţii cu jeton	Octeţi
Rata de vârf a datelor	Octeţi/sec

Dimensiunea minimă a pachetului	Octeţi
Dimensiunea maximă a pachetului	Octeţi

Fig. 5-35. Un exemplu de specificaţie a fluxului

Al doilea parametru este dimensiunea găleţii în octeţi. Dacă, de exemplu, *rata găleţii cu jeton* este de 1 Mbps şi *dimensiunea găleţii cu jeton* este de 500 KB, găleata se poate umplută continuu timp de 4 sec până să fie plină (în absenţa oricărei transmisii). Orice jeton trimis după aceasta este pierdut.

Al treilea parametru, *Rata de vârf a datelor* este rata de transmisie maximă tolerată, chiar şi pentru intervale scurte de timp. Emiţătorul nu trebuie să depăşească niciodată această valoare.

Ultimii doi parametri specifică dimensiunile minimă şi maximă ale pachetului, incluzând antetele nivelelor transport şi reţea (de exemplu TCP şi IP). Dimensiunea minimă este importantă deoarece durata procesării fiecărui pachet este fixată, indiferent cât este de mic pachetul. Un ruter poate fi pregătit să se ocupe de 10,000 de pachete/sec de câte 1 KB fiecare, dar nepregătit să se ocupe de 100000 de pachete/sec de câte 50 de octeţi fiecare, chiar dacă acestea reprezintă mai puţine date. Dimensiunea maximă a pachetului este importantă datorită limitărilor interne ale reţelei, care nu pot fi depăşite. De exemplu, dacă o parte a căii trece prin Ethernet, dimensiunea maximă a pachetului va fi restricţionată la nu mai mult de 1500 de octeţi, indiferent de cât poate să suporte restul reţelei.

O întrebare interesantă este cum transformă un ruter o specificaţie a fluxului într-un set de rezervări specifice de resurse. Această mapare este implementată specific şi nu este standardizată. Să presupunem că un ruter poate procesa 100000 pachete/sec. Dacă îi este oferit un flux de 1MB/sec cu dimensiunile minimă şi maximă a pachetului de 512 octeţi, ruterul poate calcula că ar putea lua 2048 de pachete/sec din acel flux. În acest caz, el trebuie să rezerve 2% din procesor pentru acel flux, preferabil mai mult pentru a evita întârzierile cauzate de aşteptarea în coadă. Dacă politica unui ruter este de a nu aloca niciodată mai mult de 50 % din procesor (ceea ce implică o întârziere dublă) şi este deja solicitat 49%, atunci acest flux trebuie respins. Calcule asemănătoare sunt necesare şi pentru celelalte resurse.

Cu cât este mai exigentă specificaţia, cu atât ea este mai folositoare ruterelor. Dacă o specificaţie de flux spune că are nevoie de *o rată a găleţii cu jeton* de 5MB/sec dar pachetele pot varia de la 50 de octeţi la 1500 de octeţi, atunci rata pachetelor va varia de la circa 3500 de pachete/sec la 105000 de pachete/sec. Ruterul s-ar putea panica la acest număr şi ar putea respinge fluxul, deşi la o dimensiune minimă a pachetului de 1000 de octeţi, fluxul de 5MB/sec ar fi fost acceptat.

Dirijarea proporţională

Cei mai mulţi algoritmi de dirijare încearcă să găsească cea mai bună cale pentru fiecare destinaţie şi îndreaptă tot traficul spre acea destinaţie pe cea mai bună cale. O abordare diferită care a fost propusă pentru a furniza o mai bună calitate a serviciilor este de a împărţi traficul pentru fiecare destinaţie pe mai multe căi. Din moment ce ruterele nu au în general o imagine de ansamblu completă asupra traficului din reţea, singura posibilitate reală de a împărţi traficul pe mai

multe rute este de a folosi informaţiile disponibile local. O metodă simplă este de a împărţi traficul în mod egal sau proporţional cu capacitatea legăturilor de ieşire. Totuşi, sunt disponibili şi algoritmi mai sofisticaţi (Nelakuditi şi Zhang, 2002).

Planificarea pachetelor

Dacă un ruter se ocupă de mai multe fluxuri, există pericolul ca un flux să ia prea mult din capacitate, îngheţând toate celelalte fluxuri. Procesare pachetelor în ordinea sosirii lor înseamnă că un emiţător agresiv poate acapara cea mai mare parte din capacitatea ruterelor pe care le traversează pachetele sale, micşorând calitatea serviciilor pentru celelalte. Pentru a împiedica asemenea tentative au fost inventaţi diverşi algoritmi de planificare a pachetelor (Bhatti şi Crowcroft, 2000).

Unul dintre primii a fost algoritmul **aşteptare echitabilă (fair queuing)** (Nagle, 987). Esenţa algoritmului este că ruterele au cozi separate pentru fiecare linie de ieşire, câte una pentru fiecare flux. Când o linie se eliberează, ruterul scanează cozile după metoda round robin, luând primul pachet din coada următoare. În acest fel, cu *n* calculatoare gazdă care concurează pentru o anumită linie de ieşire, fiecare gazdă reuşeşte să expedieze un pachet din fiecare *n*. Trimiterea mai multor pachete nu va îmbunătăţi rata.

Deşi este un început, algoritmul are o problemă: dă mai multă lăţime de bandă gazdelor care folosesc pachete mari decât celor care folosesc pachete mici. Demers ş.a. (1990) a sugerat o îmbunătăţire în care parcurgerea round robin este realizată astfel încât să simuleze un round-robin octet-cu- octet în locul unui round robin pachet-cu-pachet. În concluzie, el scanează cozile în mod repetat, octet cu octet, până găseşte momentul la care fiecare pachet va fi terminat. Apoi pachetele sunt sortate în ordine terminării lor şi trimise în acea ordine. Algoritmul este ilustrat în fig. 5-36.

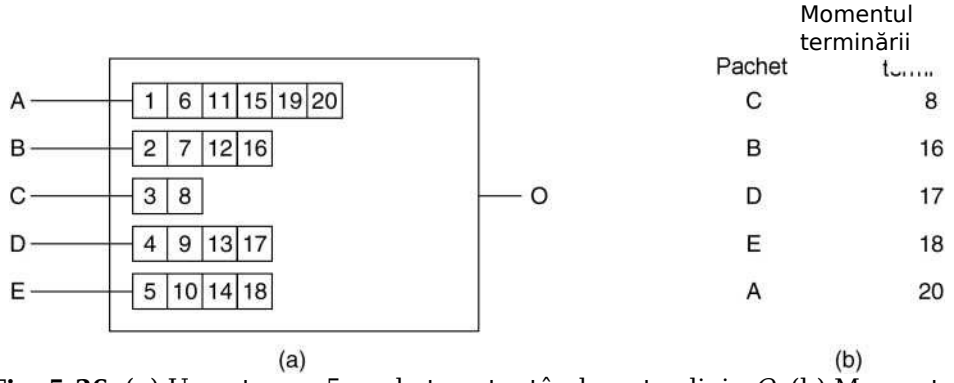


Fig. 5-36. (a) Un ruter cu 5 pachete aşteptând pentru linia *O.* (b) Momentul terminării pentru cele 5 pachete.

În fig. 5-36(a) observăm pachete cu lungimi între 2 şi 6 octeţi. La momentul (virtual) 1 este trimis primul octet al pachetului de pe linia A. Apoi urmează primul octet al pachetului de pe linia B şi aşa mai departe. Primul pachet care este terminat este C, după opt perioade. Ordinea sortată este dată în fig. 5-36(b). În absenţa unor noi sosiri, pachetele vor fi trimise în ordinea listată, de la C la A.

O problemă a acestui algoritm este aceea că acordă tuturor calculatoarelor

gazdă aceeaşi prioritate. În multe situaţii, este de dorit să se acorde mai multă lăţime de bandă serverelor video decât serverelor de fişiere normale în aşa fel încât să poată transmită doi sau mai mulţi octeţi pe perioadă. Acest algoritm modificat se numeşte **aşteptare echitabilă ponderată** (**weighted fair queueing**) şi este foarte folosit. Uneori ponderea este egală cu numărul de fluxuri care ies dintr-un calculator, astfel încât toate procesele primesc lăţime de bandă egală. O implementare eficientă a acestui algoritm este discutată în (Shreedhar şi Varghese, 1995). Din ce în ce mai mult, retransmiterea efectivă a pachetelor printr-un ruter sau comutator se face prin hardware (Elhananz et al., 2001).

5.4.3 Servicii integrate

Între 1995 şi 1997, IETF a depus mult efort pentru a inventa o arhitectură pentru fluxurile de tip multimedia. Această muncă s-a concretizat în peste doua duzini de RFC-uri, începând cu RFC-urile 2205-2210. Numele generic al acestui rezultat este **algoritmi bazaţi pe flux (flow-based algorithms)** sau **servicii integrate (integrated services)**. A fost gândit atât pentru aplicaţii cu trimitere unică (eng.: unicast) cât şi pentru cele cu trimitere multiplă (eng.: multicast). Un exemplu pentru cele dintâi este cazul unui singur utilizator rulând o secvenţă video de pe un sit de ştiri. Un exemplu pentru cel din urmă este o colecţie de staţii de televiziune prin cablu difuzându-şi programele ca şiruri de pachete IP mai multor receptori din diverse locaţii. În cele ce urmează ne vom concentra pe multicast, având în vedere că unicastul este un caz particular al multicast-ului.

În numeroase aplicaţii multicast, grupurile îşi pot schimba dinamic componenţa, de exemplu ca participanţii la o videoconferinţă care se plictisesc şi comută pe un canal cu un serial uşurel sau pe canalul de crochet. În aceste condiţii, abordarea în care emiţătorii rezervă lăţime de bandă în avans nu mai funcţionează corespunzător, deoarece va cere fiecărui emiţător să urmărească toate intrările şi ieşirile celor din audienţa sa. Pentru un sistem destinat transmisiilor televiziune cu milioane de abonaţi, nu va merge deloc.

RSVP – Protocol de rezervare a resurselor

Principalul protocol IETF pentru arhitectura serviciilor integrate este **RSVP (Resource reSerVation Protocol)**. El este descris în RFC 2205 şi altele. Acest protocol este folosit pentru a face rezervări; pentru transmisia datelor sunt folosite alte protocoale. RSVP permite emiţătorilor multipli să transmită spre grupuri multiple de receptori, permite fiecărui receptor să schimbe canalul la alegere şi optimizează lăţimea de bandă folosită, eliminând în acelaşi timp congestia.

În forma sa cea mai simplă, protocolul foloseşte dirijarea multicast cu arbori de acoperire, aşa cum s-a discutat anterior. Fiecare grup are asociată o adresă de grup. Pentru a trimite unui grup, emiţătorul pune adresa grupului în pachetele pe care le trimite. În continuare, algoritmul standard de dirijare multicast va construi un arbore de acoperire care acoperă toţi membrii grupului. Algoritmul de dirijare nu este parte a RSVP. Singura diferenţă faţă de multicast-ul normal este o mică informaţie suplimentară distribuită periodic grupului pentru a spune ruterelor de-a lungul arborelui să menţină anumite structuri de date.

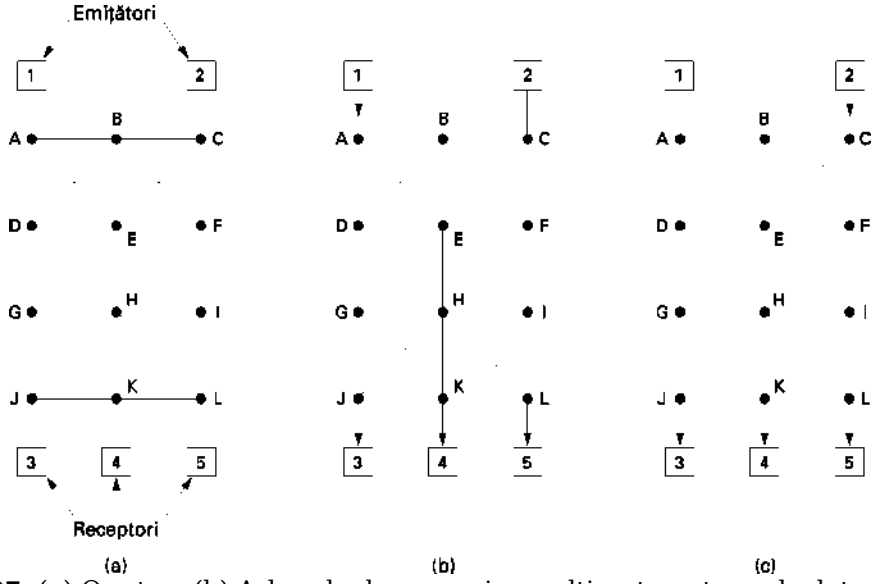


Fig. 5-37. (a) O reţea. (b) Arborele de acoperire multicast pentru calculatorul 1. (c) Arborele de acoperire multicast pentru calculatorul 2.

Ca exemplu, să considerăm reţeaua din fig. 5-37(a). Calculatoarele gazdă 1 şi 2 sunt emiţători multicast, iar calculatoarele gazdă 3, 4 şi 5 sunt receptori multicast. În acest exemplu emiţătorii şi receptorii sunt disjuncţi, dar în general cele două mulţimi se pot suprapune. Arborii multicast pentru maşinile 1 şi 2 sunt prezentaţi în fig. 5-37(b), respectiv fig. 5-37(c).

Pentru a avea o recepţie mai bună şi pentru a elimina congestia, fiecare dintre receptorii dintr-un grup poate să trimită un mesaj de rezervare în sus pe arbore spre emiţător. Mesajul este propagat folosind algoritmul căii inverse discutat anterior. La fiecare salt, ruterul notează rezervarea şi rezervă lăţimea de bandă necesară. Dacă lăţimea de bandă este insuficientă, se raportează eroare. Atunci când mesajul ajunge la sursă, lăţimea de bandă a fost rezervată pe tot drumul de la emiţător spre receptorul care a făcut rezervarea de-a lungul arborelui de acoperire.

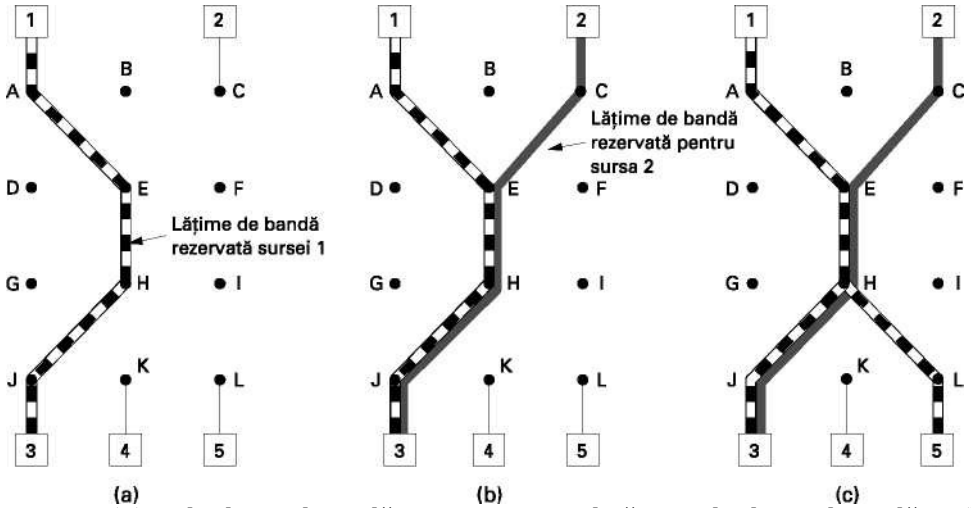


Fig. 5-38. (a) Calculatorul gazdă 3 cere un canal către calculatorul gazdă 1. (b) Calculatorul gazdă 3 cere un al doilea canal către calculatorul gazdă 2. (c) Calculatorul gazdă 5 cere un canal către calculatorul gazdă 1.

Un exemplu de astfel de rezervare este prezentat în fig. 5-38(a). Aici calculatorul gazdă 3 a cerut un canal către calculatorul gazdă 1. Odată ce acesta a fost stabilit, pachetele pot ajunge de la 1 la 3 fără congestie. Să vedem acum ce se întâmplă dacă gazda 3 rezervă şi un canal către celălalt emiţător, calculatorul gazdă 2, astfel încât utilizatorul să poată urmări două posturi de televiziune simultan. Se rezervă o a doua cale, aşa cum se ilustrează în fig. 5-38(b). Observaţi că sunt necesare două canale distincte de la calculatorul gazdă 3 către ruterul *E,* deoarece se transmit două fluxuri independente.

În fine, în fig. 5-38 (c) calculatorul gazdă 5 decide să urmărească programul de televiziune transmis de 1 şi face de asemenea o rezervare. Pentru început, se rezervă lăţime de bandă spre ruterul *H.* Acest ruter observă că are deja o rezervare către 1, astfel încât dacă lăţimea de bandă necesară a fost rezervată, nu mai trebuie să rezerve nimic. Se poate întâmpla ca 3 şi 5 să ceară lăţimi de bandă diferite (de exemplu, 3 are un sistem de televiziune alb-negru, astfel încât nu are nevoie de informaţia de culoare), caz în care capacitatea rezervată trebuie să fie suficientă pentru a satisface cererea cea mai mare.

La realizarea unei rezervări, un receptor poate (opţional) specifica una sau mai multe surse de la care vrea să recepţioneze. De asemenea, el mai poate specifica dacă aceste alegeri sunt fixe pe durata rezervării sau dacă receptorul doreşte să-şi păstreze opţiunea de a modifica sursele ulterior. Ruterele folosesc această informaţie pentru a optimiza planificarea lăţimii de bandă. În particular, doi receptori pot să partajeze o cale doar dacă ambii stabilesc să nu modifice sursele ulterior.

Motivul acestei strategii în cazul dinamic sută la sută este că rezervarea lăţimii de bandă este decuplată de alegerea sursei. Odată ce un receptor a rezervat lăţime de bandă, el poate comuta către altă sursă şi să păstreze porţiunea din calea existentă care este comună cu calea către noua sursă. Când calculatorul 2 transmite mai multe fluxuri video, de exemplu, calculatorul 3 poate comuta între ele după

dorinţă, fără a-şi schimba rezervarea: ruterelor nu le pasă la ce program se uită utilizatorul.

5.4.4 Servicii diferenţiate

Algoritmii bazaţi pe flux au potenţialul de a oferi o bună calitate a serviciilor unuia sau mai multor fluxuri deoarece acestea îşi rezervă resursele necesare de-a lungul rutei. Totuşi ei au şi un dezavantaj: au nevoie să stabilească în avans caracteristicile fiecărui flux, ceea ce nu este optim în cazul în care există milioane de fluxuri. De asemenea, ei memorează caracteristicile fiecărui flux ca date interne ale ruterului, ceea ce le face vulnerabile în cazul căderii ruterului. În fine, modificările necesare codului ruterului sunt substanţiale şi implică schimburi complexe de informaţii între rutere pentru stabilirea fluxurilor. Ca o consecinţă, există foarte puţine implementări ale RSVP-ului sau a ceva asemănător.

Din aceste motive, IETF a propus şi o abordare mai simplă a calităţii serviciilor, care poate fi implementată local în fiecare ruter fără iniţializări prealabile şi fără a implica întreaga cale. Această abordare este cunoscută sub numele de calitatea serviciilor **orientate pe clase (class-based)** (opusă abordării bazate pe flux). IETF a standardizat o arhitectură numită **servicii diferenţiate** (**differentiated services**), care este descrisă în RFC-urile 2474, 2475 şi altele. O vom descrie în continuare.

Serviciile diferenţiate (DS) pot fi oferite de către un set de rutere care formează un domeniu administrativ (de exemplu un ISP sau telco). Administraţia defineşte un set de clase de servicii cu regulile de rutare corespunzătoare. Dacă un client se înscrie pentru DS, pachetele clientului care intră în domeniu pot avea un câmp *Type of Service (Tipul serviciului),* cu servicii mai bine furnizate unor clase (de exemplu serviciu premium) decât altora. Traficului dintr-o clasă i se poate cere să se conformeze unei anumite forme, cum ar fi găleata găurită cu o anumită rată decurgere. Un operator cu fler ar putea să ceară mai mult pentru fiecare pachet premium transportat sau ar putea să permită maximum N pachete premium pe lună pentru o taxă lunară suplimentară fixă. Observaţi că această schemă nu presupune iniţializarea în avans, nici rezervarea resurselor şi nici negocierea capăt-la- capăt pentru fiecare flux, care consumă timp, ca în cazul serviciilor integrate. Aceasta face ca serviciile diferenţiate să fie relativ uşor de implementat.

Serviciile bazate pe clase se întâlnesc şi în alte domenii. De exemplu, companiile care livrează pachete oferă de obicei servicii de tip “peste noapte”, “în două zile” sau “în trei zile”. Companiile aeriene oferă clasa întâi, clasa de afaceri şi clasa a doua. Trenurile pe distanţe lungi oferă adeseori multiple clase de servicii. Chiar şi metroul din Paris are două tipuri de servicii. Pentru pachete, clasele pot să difere, printre altele, prin valorile întârzierii, fluctuaţiei şi a probabilităţii pachetului de a fi aruncat la apariţia unei congestii.

Pentru a evidenţia diferenţa dintre calitatea serviciilor bazate pe flux şi calitatea serviciilor bazate pe clase, să considerăm un exemplu: telefonia Internet. În cazul organizării pe flux, fiecare apel telefonic are propriile resurse şi garanţii. În cazul organizării pe clase, toate apelurile telefonice beneficiază de resursele rezervate pentru clasa telefoniei. Aceste resurse nu pot fi preluate de pachete din clasa transferului de fişiere sau din alte clase, dar nici un apel telefonic nu beneficiază de resurse particulare, rezervate doar pentru acesta.

Retransmitere expeditivă (Expedited Forwarding)

Alegerea claselor de servicii este la dispoziţia fiecărui operator, dar, având în vedere că pachetele sunt adesea expediate între subreţele administrate de operatori diferiţi, IETF lucrează la definirea unor clase de servicii independente de reţea. Cea mai simplă clasă este **retransmiterea expeditivă (expedited forwarding)**, deci vom începe cu aceasta. Descrierea ei este dată în RFC 3246.

Ideea care stă la baza retransmiterii expeditive este foarte simplă. Sunt disponibile două clase de servicii: normale şi rapide (expeditive). Marea majoritate a traficului se presupune că este de tip normal, dar o mică parte se face şi expeditiv. Pachetele din această clasă ar trebui să poată traversa subreţeaua ca şi cum nu ar mai exista şi alte pachete. O reprezentare simbolică a acestui sistem "bicanal" este dată în fig. 5-39. Observaţi că există doar o singură linie fizică. Cele două canale logice din figură reprezintă o cale de a rezerva lăţime de bandă şi nu o a doua linie fizică.

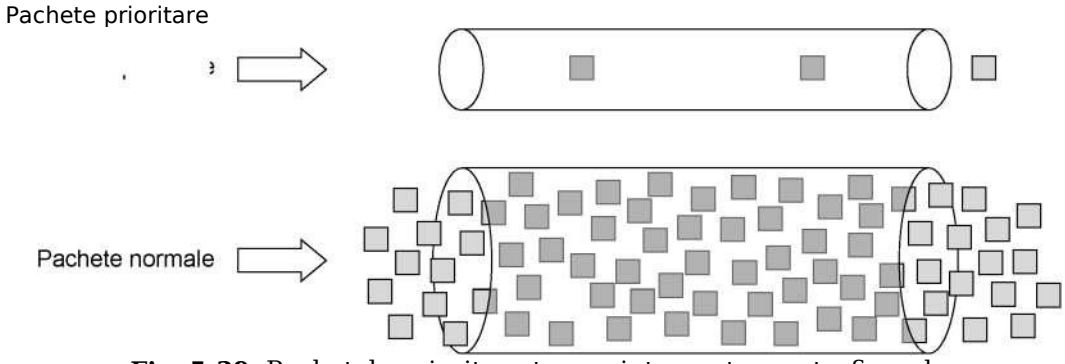


Fig. 5-39. Pachetele prioritare trec printr-o reţea cu trafic redus.

O modalitate de a implementa această strategie este de a programa ruterele astfel încât să aibă două cozi de aşteptare pentru fiecare linie de ieşire, una pentru pachete prioritare şi cealaltă pentru pachete normale. Când soseşte un pachet este introdus în coada corespunzătoare. Planificarea pachetelor ar trebui să utilizeze ceva de genul aşteptării echitabile ponderate. De exemplu, dacă 10% din trafic este de tip expeditiv şi 90% de tip normal, atunci 20% din lăţimea de bandă ar putea fi repartizată traficului expeditiv, iar restul traficului normal. În acest fel traficul expeditiv ar avea de două ori mai multă lăţime de bandă decât îi este necesar pentru a asigura întârziere mică . Această repartizare poate fi obţinută transmiţând câte un pachet prioritar la fiecare patru pachete normale (presupunând că distribuţia dimensiunilor pachetelor ambelor clase este similară). În acest fel se speră că pachetele prioritare nu văd subreţeaua ca fiind aglomerată, chiar dacă există o încărcare substanţială.

Rutare garantată

O metodă mai elaborată de a administra clasele de servicii este **rutarea garantată (assured forwarding)**. Descrierea acesteia se găseşte în RFC 2597. Ea precizează că ar trebui să existe patru clase de priorităţi, fiecare cu propriile resurse. În plus defineşte trei probabilităţi de aruncare a pachetelor care sunt supuse congestiei: mică, medie şi mare. Luate împreună, aceste două criterii defi-

nesc 12 clase de servicii.

Fig. 5-40 prezintă o modalitate în care pachetele ar putea fi procesate în cazul rutării garantate. Primul pas este acela de a repartiza pachetele într-una din cele patru clase de priorităţi. Acest pas se poate face pe calculatorul emiţător (după cum se vede şi din figură) sau în primul ruter. Avantajul realizării clasificării pe calculatorul gazdă care realizează transmisia este acela că are la dispoziţie mai multe informaţii despre fiecare pachet şi direcţia în care se îndreaptă .

Al doilea pas este de a marca pachetele în conformitate cu clasa din care fac parte. Pentru aceasta este necesar un câmp antet. Din fericire în antetul IP este disponibil un câmp pe 8 biţi numit *Tipul serviciului,* după cum vom vedea în continuare. RFC 2597 specifică faptul că şase dintre aceşti biţi se vor folosi pentru clasa de serviciu, lăsând loc pentru codificarea claselor istorice şi a celor viitoare.

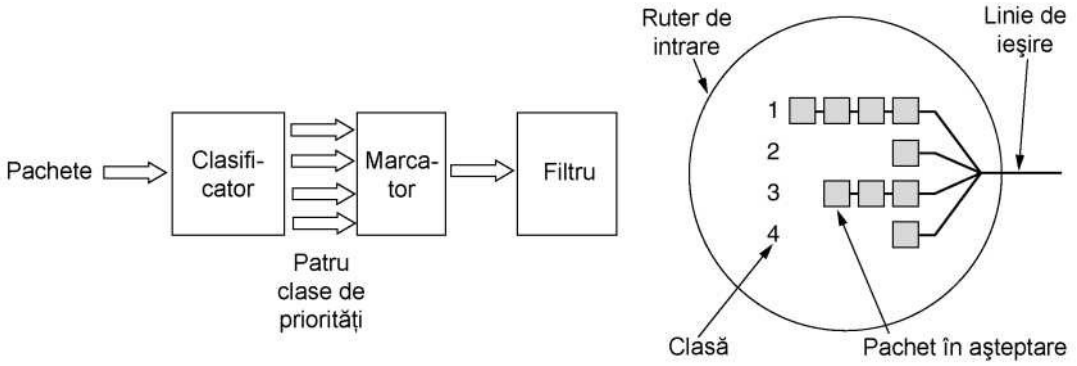


Fig. 5-40. O posibilă implementare a fluxului de date pentru rutarea garantată.

Pasul al treilea presupune trecerea pachetelor printr-un filtru (eng.: shaper/dropper filter) care ar putea întârzia sau arunca o parte dintre ele pentru a forma cele patru fluxuri într-o manieră acceptabilă, folosind de exemplu algoritmul găleţii găurite sau al găleţii cu jeton. Dacă sunt prea multe pachete, unele dintre ele ar putea fi aruncate în această etapă, în funcţie de categoria în care au fost plasate de filtru. Sunt posibile şi scheme mai elaborate care implică măsurători şi reacţii inverse

În acest exemplu, paşii specificaţi sunt efectuaţi pe calculatorul emiţător, deci fluxul rezultat este trimis primului ruter. Merită observat faptul că aceşti paşi pot fi efectuaţi de un software de reţea specializat sau chiar de către sistemul de operare, pentru a evita modificarea aplicaţiei existente.

5.4.5 Comutarea etichetelor şi MPLS

În timp ce IETF lucra la serviciile integrate şi diferenţiate, unii dintre vânzătorii de rutere căutau metode mai bune de rutare. Munca lor s-a axat pe adăugarea unei etichete în faţa fiecărui pachet şi pe efectuarea rutării mai degrabă pe baza acestei etichete decât a adresei destinaţie. Utilizarea etichetei ca index într-o tabelă internă a ruterului face ca găsirea liniei de ieşire corecte să se transforme într-o simplă căutare în tabel. Folosind această metodă, rutarea se poate face foarte repede iar resursele necesare pot fi rezervate pe traseu.

Bineînţeles, etichetarea fluxurilor în acest fel se apropie foarte mult de circuitele virtuale. X.25, ATM, releu de cadre (eng.: frame-relay) şi toate celelalte reţele cu subreţele cu circuite virtuale pun şi ele câte o etichetă (identificatorul circuitului virtual) în fiecare pachet, o caută într-o tabelă şi rutarea se face pe baza intrării din tabelă. În ciuda faptului că mulţi dintre membrii comunităţii Internet au o puternică antipatie faţă de rutarea orientată pe conexiune, se pare că nu se renunţă la idee, de data aceasta pentru a furniza o rutare rapidă şi calitatea serviciilor. Totuşi există diferenţe fundamentale între modul în care se ocupă Internetul de construcţia rutelor şi modul în care o fac reţelele orientate pe conexiune, deci în mod sigur tehnica nu este tradiţionala comutare de circuite.

"Noua" idee de comutare este cunoscută sub mai multe denumiri, inclusiv **comutarea etichetelor** (eng.: **label switching**) şi **comutarea marcajelor** (eng.: **tag switching**). În cele din urmă, IETF a început să standardizeze ideea sub numele de **multiprotocol de comutare a etichetelor (MPLS – MultiProtocol Label Switching)**. În continuare o vom numi MPLS. Aceasta este descrisă în RFC 3031 şi în multe alte RFC-uri.

Pe de altă parte, unii fac diferenţa între *rutare* şi *comutare.* Rutarea este procesul de căutare a adresei destinaţie în tabel pentru a găsi unde trebuie trimis. În schimb, comutarea foloseşte o etichetă luată dintr-un pachet ca index într-o tabelă de rutare. Aceste definiţii sunt totuşi departe de a fi universale.

Prima problemă care apare este unde se pune eticheta. Din moment ce pachetele IP nu au fost proiectate pentru circuite virtuale, în cadrul antetelor IP nu există nici un câmp disponibil pentru numerele circuitelor virtuale. Din acest motiv, în faţa antetului IP a trebuit adăugat un nou antet MPLS. Pe o linie ruter-la-ruter folosind PPP ca protocol de încadrare, formatul cadrului, inclusiv
antetele PPP, MPLS, IP şi TCP, arată ca în fig. 5-41. Într-un fel, MPLS este nivelul 2.5.

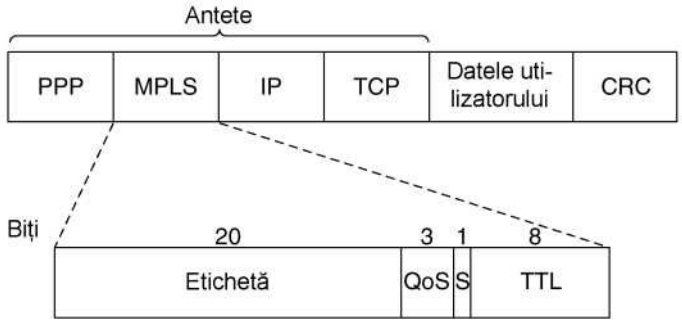


Fig. 5-41. Transmiterea unui segment TCP utilizând IP, MPLS şi PPP.

Antetul generic MPLS are 4 câmpuri, dintre care cel mai important este câmpul *Etichetă (Label)* care deţine indexul. Câmpul *QoS (Quality of Service; rom.: Calitatea serviciilor)* precizează clasa de servicii. Câmpul *S* se referă la memorarea mai multor etichete în reţelele ierarhice (care vor fi discutate mai târziu). Dacă ajunge la valoarea 0, pachetul este aruncat. Această facilitate previne buclarea infinită în cazul instabilităţii rutării.

Deoarece antetele MPLS nu fac parte din pachetul format la nivelul reţea şi nici din cadrele nivelului legătură de date, MPLS reprezintă o extindere a acestor două

nivele. Printre altele aceasta înseamnă că este posibilă construirea de comutatoare MPLS care să poată ruta atât pachete IP cât şi celule ATM, după necesităţi. Această caracteristică a dus la apariţia termenului "multiprotocol" din denumirea de MPLS.

Când un pachet îmbunătăţit MPLS (sau o celulă) ajunge la un ruter ce suportă MPLS, eticheta este folosită ca index într-un tabel pentru a determina linia de ieşire ce va fi folosită precum şi noua etichetă. Această schimbare de etichete se foloseşte în toate subreţelele cu circuite virtuale deoarece etichetele au doar semnificaţie locală şi două rutere diferite pot trimite pachete care nu au legătură unul cu altul dar au aceeaşi etichetă unui alt ruter pentru a fi transmise pe aceeaşi linie de ieşire. Pentru a putea fi diferenţiate la capătul celălalt, etichetele trebuie să fie remapate la fiecare salt. Am văzut cum funcţionează acest mecanism în fig. 5-3. MPLS foloseşte aceeaşi tehnică.

O diferenţă faţă de circuitele virtuale tradiţionale este nivelul de agregare. În mod sigur este posibil ca fiecare flux să aibă propriul set de etichete în cadrul subreţelei. Totuşi se obişnuieşte mai mult ca ruterele să grupeze mai multe fluxuri care au ca destinaţie un anumit ruter sau LAN şi să folosească pentru ele o singură etichetă. Despre fluxurile care sunt grupate sub o aceeaşi etichetă se spune că aparţin aceleiaşi **Clase echivalente de rutare (FEC – Forwarding Equivalence Class)**. Această clasă acoperă nu numai pachetele care pleacă, dar şi clasele lor de servicii (în sensul claselor diferenţiate de servicii) deoarece toate pachetele pe care le conţin sunt tratate la fel din motive de expediere.

La rutarea tradiţională de tip circuit virtual nu este posibil să se grupeze mai multe căi cu destinaţii diferite pe acelaşi identificator de circuit virtual, deoarece la destinaţia finală nu ar mai fi posibil să se facă distincţie între ele. Cu MPLS, pachetele conţin, pe lângă etichetă, şi adresa lor finală de destinaţie, în aşa fel încât, la capătul rutei etichetate, antetul poate fi eliminat şi rutarea poate continua în mod obişnuit, folosind adresa de destinaţie a nivelului reţea.

O diferenţă majoră dintre MPLS şi circuitele virtuale convenţionale este modul în care este construită tabela de rutare. În reţelele cu circuite virtuale tradiţionale, când un utilizator doreşte să stabilească o conexiune, este lansat în reţea un pachet de iniţializare pentru a crea calea şi intrările tabelelor de rutare. MPLS nu funcţionează astfel deoarece nu există fază de iniţializare pentru fiecare conexiune (deoarece ar stopa prea mult din software-ul existent în Internet).

În schimb există două modalităţi în care pot fi create intrările tabelelor de rutare. În abordarea **bazată pe date**, **(data-driven)** atunci când un pachet ajunge la primul ruter, acesta contactează ruterul din aval, la care trebuie să ajungă pachetul, şi îi cere să genereze o etichetă pentru flux. Această metodă se aplică recursiv. Practic, aceasta este crearea circuitelor virtuale la cerere.

Protocoalele care realizează această răspândire au foarte mare grijă să evite buclele. Ele folosesc adeseori tehnica numită a **firelor colorate (colored threads)**. Propagarea înapoi a unei FEC poate fi comparată cu tragerea unui fir de o singură culoare înapoi în subreţea. Dacă un ruter întâlneşte o culoare pe care o are deja, el ştie că există o buclă şi ia măsuri pentru remedierea situaţiei. Abordarea bazată pe date este folosită în reţelele unde sub nivelul transport se găseşte ATM-ul (ca în cazul sistemului telefonic).

Cealaltă modalitate, folosită în reţele care nu sunt bazate pe ATM, este

abordarea bazată pe control (eng.: **control-driven**). Ea are mai multe variante, dintre care una funcţionează în felul următor. Când este pornit un ruter, verifică pentru ce rute el reprezintă destinaţia finală (de exemplu ce calculatoare gazdă sunt în reţeaua locală). Apoi creează una sau mai multe clase echivalente pentru acestea, alocă pentru fiecare câte o etichetă şi trimite etichetele vecinilor săi. Aceştia, la rândul lor, introduc etichetele în tabelele de rutare şi trimit noi etichete vecinilor lor, până când toate ruterele şi-au însuşit calea. De asemenea, pentru a garanta o calitate corespunzătoare a serviciilor, resursele pot fi alocate pe măsura construirii căii.

MPLS poate opera la mai multe niveluri deodată. La nivelul cel mai înalt, fiecare companie de telecomunicaţii poate fi privită ca un fel de metaruter, existând o cale prin metarutere de la sursă la destinaţie. Această cale poate folosi MPLS. Totuşi, în cadrul reţelei fiecărei companii de telecomunicaţii poate fi folosit de asemenea MPLS, ducând la un al doilea nivel de etichetare. De fapt, un pachet poate transporta o întreagă stivă de etichete. Bitul *S* din fig. 5-41 permite unui ruter care şterge o etichetă să afle dacă au mai rămas alte etichete. Acesta are valoarea 1 pentru eticheta de la bază şi 0 pentru toate celelalte etichete. În practică, această facilitate este folosită în principal la implementarea reţelelor virtuale private şi a tunelelor recursive.

Deşi ideile care au stat la baza MPLS-ului sunt simple, detaliile sunt extrem de complicate, cu multe variaţii şi optimizări, aşa că nu vom mai dezvolta acest subiect. Pentru mai multe informaţii, vezi (Davie şi Rekhter, 2000; Lin et al., 2002; Pepelnjak şi Guichard, 2001; şi Wang, 2001).

5.5 INTERCONECTAREA REŢELELOR

Până în acest moment, am presupus implicit că există o singură reţea omogenă, în care fiecare maşină foloseşte acelaşi protocol la fiecare nivel. Din păcate, această presupunere este prea optimistă. Există mai multe tipuri de reţele, incluzând LAN-uri, MAN-uri şi WAN-uri. La fiecare nivel se

folosesc pe larg numeroase protocoale. În secţiunile următoare vom analiza în detaliu problemele care apar când două sau mai multe reţele sunt conectate pentru a forma o **inter-reţea** (internet).

Există controverse considerabile pe tema întrebării dacă abundenţa actuală de tipuri de reţele este o condiţie temporară, care va dispărea de îndată ce toată lumea îşi va da seama ce minunată este [completaţi cu tipul preferat de reţea], sau dacă este o caracteristică inevitabilă, dar permanentă, a lumii care va persista. Existenţa reţelelor de tipuri diferite înseamnă, invariabil, a avea protocoale diferite.

Credem că întotdeauna vor coexista o varietate de reţele diferite (şi implicit de protocoale) din următoarele motive. În primul rând, numărul de reţele diferite instalate este mare. Aproape toate calculatoarele personale folosesc TCP/IP. Multe întreprinderi mari încă se bazează pe sisteme de calcul care folosesc SNA de la IBM. O parte importantă din companiile telefonice operează reţele ATM. Unele LAN-uri de calculatoare personale folosesc încă Novell NCP/IPX sau AppleTalk. Şi, în final, în domeniul în plină dezvoltare al comunicaţiilor fără fir există o mare varietate de protocoale. Această tendinţă va continua mult timp datorită problemelor de continuitate, noilor tehnologii şi faptului că nu toţi producătorii percep a fi în interesul lor posibilitatea clienţilor de a migra cu uşurinţă spre sistemul altui producător.

În al doilea rând, pe măsură ce calculatoarele şi reţelele devin mai ieftine, nivelul la care se iau deciziile se mută în jos în organizaţie. Multe companii au o politică care stabileşte că achiziţiile de peste un milion de dolari trebuie să fie aprobate de conducerea superioară, achiziţiile de peste 100.000 de dolari trebuie să fie aprobate de conducerea medie, dar achiziţiile de până la 100.000 de dolari pot fi făcute de şefii de departamente fără nici o aprobare de mai sus. Aceasta poate duce uşor la instalarea unor staţii de lucru UNIX care rulează TCP/IP în departamentul inginerie şi a unor calculatoare Macintosh care folosesc AppleTalk în departamentul de marketing.

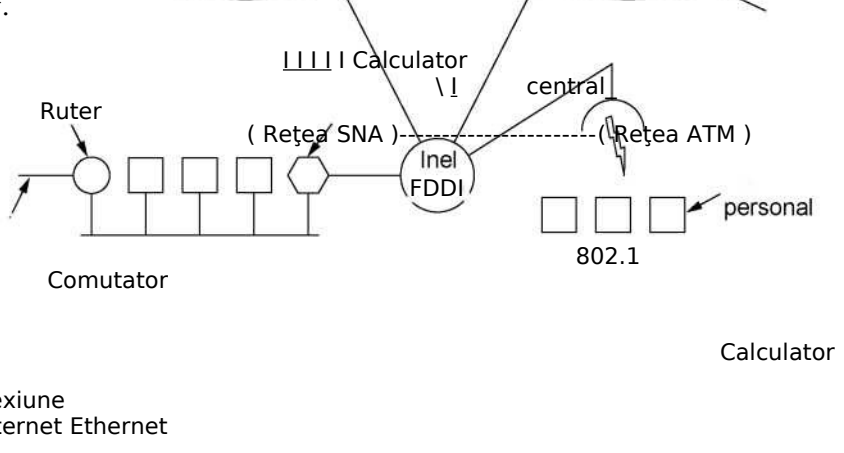


Fig. 5-42. Colecţie de reţele interconectate.

În al treilea rând, reţele diferite (de exemplu ATM şi fără fir) sunt bazate pe tehnologii radical diferite şi nu ar trebui să surprindă că pe măsură ce apar dezvoltări hardware, se vor crea noi programe care să se potrivească cu noul

hardware. De exemplu, casa medie de astăzi este asemănătoare cu biroul mediu de acum 10 ani; este plină de calculatoare care nu comunică unul cu celălalt. În viitor, ar putea fi un lucru obişnuit ca telefonul, televizorul şi alte aparate electrocasnice să fie legate în reţea, pentru a permite controlul de la distanţă . Această nouă tehnologie va aduce, fără dubii, noi reţele şi noi protocoale.

Ca un exemplu al modului în care ar putea fi conectate reţele diferite, să considerăm exemplul din fig. 5-42. Aici vedem reţeaua unei companii cu mai multe sedii interconectate printr-o reţea ATM de mare întindere. La unul dintre sedii este folosită o coloană vertebrală (eng.: backbone) optică FDDI, prin care sunt conectate: o reţea Ethernet, un LAN fără fir 802.11 şi reţeaua SNA de calculatoare centrale a centrului de date al corporaţiei.

Scopul interconectării acestor reţele este de a permite utilizatorilor din orice reţea să comunice cu utilizatorii celorlalte reţele şi de asemenea de a permite unui utilizator din orice reţea să acceseze date pe orice reţea. Realizarea acestui scop înseamnă trimiterea pachetelor dintr-o reţea în alta. Cum reţelele diferă deseori în puncte esenţiale, transmiterea pachetelor dintr-o reţea în alta nu este întotdeauna uşoară, după cum vom vedea în continuare.

5.5.1 Prin ce diferă reţelele

Reţelele pot diferi în multe moduri. Unele dintre deosebiri, cum ar fi diferite tehnici de modulare sau formate de cadre, se găsesc la nivelul fizic şi legătură de date. Aceste diferenţe nu ne preocupă la acest nivel. În schimb, în fig. 5-43, enumerăm câteva din diferenţele care pot apărea la nivelul reţea. Trecerea peste aceste diferenţe face interconectarea reţelelor mult mai dificilă decât operarea într-o singură reţea.

Element	**Câteva posibilităţi**
Serviciu oferit	Orientat pe conexiuni faţă de cel fără conexiune
Protocol	IP, IPX, SNA, ATM, MPLS, AppleTalk etc.
Adresare	Plată (802) opusă celei ierarhice (IP)
Trimitere multiplă	Prezentă sau absentă (de asemenea, difuzarea totală)
Dimensiune pachet	Fiecare reţea îşi are propriul maxim
Calitatea serviciului	Poate fi prezentă sau absentă; multe tipuri diferite
Tratarea erorilor	Livrare fiabilă, ordonată sau neordonată
Controlul fluxului	Fereastră glisantă, controlul ratei, altele sau nimic
Controlul congestiei	Algoritmul găleţii găurite, algoritmul găleţii cu jeton, RED, pachete
Securitate	Reguli de secretizare, criptare etc.
Parametri	Diferite limitări de timp, specificări ale fluxului etc.
Contabilizare	După timpul de conectare, după pachet, după octeţi sau fără.

Fig. 5-43. Câteva din multele moduri în care pot diferi reţelele.

Când pachetele trimise de o sursă dintr-o reţea trebuie să tranziteze una sau mai multe reţele străine înainte de a ajunge în reţeaua destinaţie (care, de asemenea, poate fi diferită faţă de reţeaua sursă), pot apărea multe probleme la interfeţele dintre reţele. Pentru început, atunci când pachetele dintr-o reţea orientată pe conexiuni trebuie să tranziteze o reţea fără conexiuni, poate interveni o reordonare a acestora, lucru la care emiţătorul nu se aşteaptă şi căruia receptorul nu este pregătit să-i facă faţă. Vor fi necesare frecvente conversii de protocol, care pot fi dificile dacă funcţionalitatea cerută nu poate fi exprimată. De asemenea, vor fi necesare conversii de adresă, ceea ce poate cere un sistem de catalogare. Trecerea pachetelor cu trimitere multiplă printr-o reţea care nu oferă trimitere multiplă necesită generarea de pachete separate pentru fiecare destinaţie.

Diferenţa dintre dimensiunile maxime ale pachetelor folosite de diferite reţele poate produce mari neplăceri. Cum veţi trece un pachet de 8000 de octeţi printr-o

reţea a cărei dimensiune maximă este de 1500 de octeţi? Diferenţele în calitatea serviciilor sunt o problemă în momentul în care un

pachet care are constrângeri de livrare în timp real traversează o reţea care nu oferă nici o garanţie de timp real.

Controlul erorilor, al fluxului şi al congestiei diferă frecvent între reţele diferite. Dacă sursa şi destinaţia aşteaptă amândouă ca toate pachetele să fie livrate în ordine fără erori, dar o reţea intermediară elimină pachete ori de câte ori întrevede congestie la orizont, multe aplicaţii se vor comporta neprevăzut. Diferenţele în ceea ce priveşte mecanismele de securitate, stabilirea parametrilor, regulile de contabilizare şi chiar legile naţionale referitoare la secrete pot, de asemenea, cauza probleme.

5.5.2 Cum pot fi conectate reţelele

Reţelele pot fi interconectate prin diferite dispozitive, aşa cum s-a arătat în cap.4. Să revedem pe scurt acest material. La nivelul fizic, reţelele pot fi conectate prin repetoare sau noduri (eng.: hubs), care doar transferă biţii între două reţele identice. Acestea sunt în marea lor majoritate dispozitive analogice şi nu cunosc protocoalele numerice (doar regenerează semnale).

Cu un nivel mai sus întâlnim punţile şi comutatoarele, care operează la nivelul legăturii de date. Acestea acceptă cadre, examinează adresele MAC şi retransmit cadrele către o reţea diferită, efectuând traduceri de protocol minore, ca de exemplu de la Ethernet la FDDI sau la 802.11.

La nivelul reţea avem rutere care pot conecta două reţele. Dacă două reţele au niveluri reţea diferite, ruterul poate fi capabil să transforme formatul pachetelor, cu toate că astfel de situaţii sunt din ce în ce mai rare. Un ruter care poate trata mai multe protocoale este numit **ruter multiprotocol** (eng. multiprotocol router).

La nivelul transport întâlnim porţi de transport, care por realiza interfaţa între două conexiuni de transport. De exemplu, o poartă de transport poate permite pachetelor să treacă între o reţea TCP şi o reţea SNA, care are un protocol de transport diferit, făcând legătura între o conexiune TCP şi o conexiune SNA.

În sfârşit, la nivelul aplicaţie, porţile de aplicaţie traduc semnificaţia mesajelor. De exemplu, punţile dintre poşta electronică din Internet (RFC 822) şi poşta electronică X 400 trebuie să analizeze mesajele şi să schimbe diferite câmpuri din antete.

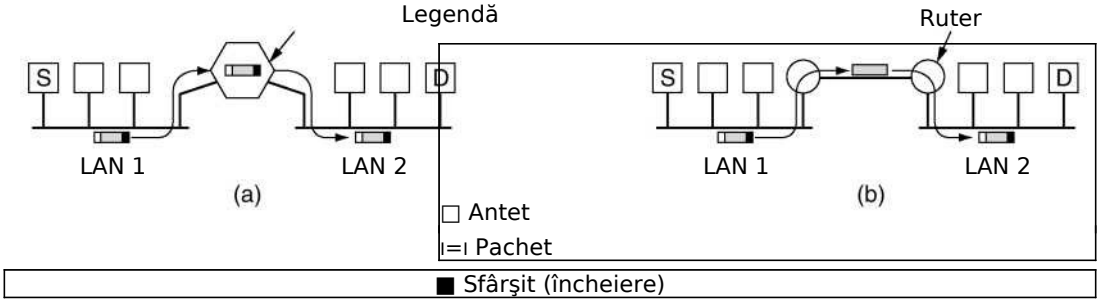


Fig. 5-44. (a) Două reţele Ethernet conectate printr-un comutator. (b) Două reţele Ethernet conectate prin rutere.

În acest capitol ne vom concentra pe interconectarea reţelelor la nivelul reţea. Pentru a vedea prin ce diferă de comutarea la nivelul legăturii de date, să examinăm fig. 5-44. În fig. 5-44(a) maşina

sursă, *S,* doreşte să trimită un pachet maşinii destinaţie, *D.* Aceste maşini se află în reţele Ethernet diferite, conectate printr-un comutator. *S* încapsulează pachetul într-un cadru şi îl trimite spre destinaţie. Cadrul ajunge la comutator, care determină, analizând adresa MAC, destinaţia cadrului, reprezentată de LAN 2. Comutatorul nu face decât să preia cadrul din LAN 1 şi să îl pună pe LAN 2.

Să considerăm acum aceeaşi situaţie, dar cu două reţele Ethernet conectate printr-o pereche de rutere, în loc de comutator. Ruterele sunt conectate printr-o linie punct-la-punct, posibil o linie închiriată de mai mulţi kilometri lungime. Acum cadrul este preluat de ruter, iar pachetul este extras din câmpul de date al cadrului. Ruterul examinează adresa din pachet (de exemplu o adresă IP) şi o caută în tabela sa de rutare. Pe baza acestei adrese decide să trimită pachetul la ruterul aflat la distanţă, eventual încapsulat într-un alt tip de cadru, în funcţie de protocolul liniei. La celălalt capăt pachetul este pus în câmpul de date al unui cadru Ethernet şi este depus pe LAN 2.

O diferenţă esenţială între cazul cu comutator (sau punte) şi cazul cu rutere este următoarea. În cazul cu comutator (sau punte) este transportat întregul cadru, pe baza adresei MAC. În cazul unui ruter pachetul este extras din cadru, iar adresa din pachet este utilizată pentru a decide unde să fie trimis. Comutatoarele nu trebuie să înţeleagă protocolul nivelului reţea, dar ruterele da.

5.5.3 Circuite virtuale concatenate

Sunt posibile două stiluri de interconectare a reţelelor: o concatenare orientată pe conexiuni a subreţelelor cu circuite virtuale şi un stil datagrame inter-reţea. Vom examina pe rând aceste variante, dar înainte de aceasta, un avertisment. În trecut, marea majoritate a reţelelor (publice) erau orientate pe conexiune (şi releul de cadre, SNA, 802.16 şi ATM încă sunt). Apoi, odată cu acceptarea rapidă a Internetului, datagramele au venit la modă. Totuşi ar fi o greşeală să credem că datagramele vor fi folosite la nesfârşit. În acest domeniu singurul lucru etern este schimbarea. Odată cu creşterea importanţei reţelelor multimedia, este probabil ca orientarea pe conexiune să revină într-o formă sau alta, deoarece este mai uşor să se garanteze calitatea serviciului cu conexiuni, decât fără ele. De aceea în continuare vom dedica spaţiu orientării pe conexiune.

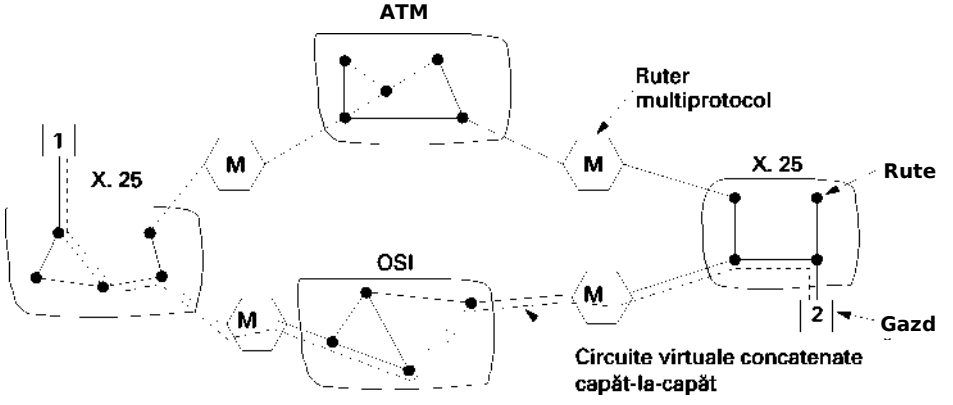


Fig. 5-45. Interconectarea reţelelor folosind circuite virtuale

În modelul circuitelor virtuale concatenate, arătat în fig. 5-45, o conexiune către o gazdă dintr-o reţea îndepărtată este stabilită într-un mod similar cu modul în care sunt stabilite conexiunile în cazul normal. Subreţeaua observă că destinaţia este îndepărtată şi construieşte un circuit virtual spre ruterul aflat cel mai aproape de reţeaua destinaţie. Apoi construieşte un circuit virtual de la acel ruter la o **poartă** externă (un ruter multiprotocol). Această poartă înregistrează existenţa circuitului virtual în tabelele sale şi continuă să construiască un alt circuit virtual către un ruter din următoarea subreţea. Acest proces continuă până când se ajunge la gazda destinaţie.

O dată ce pachetele de date încep să circule de-a lungul căii, fiecare poartă retransmite pachetele primite, făcând, după nevoie, conversia între formatele pachetelor şi numerele de circuite virtuale. Evident, toate pachetele de date trebuie să traverseze aceeaşi secvenţă de porţi, deci ajung în ordine.

Caracteristica esenţială a acestei abordări este că se stabileşte o secvenţă de circuite virtuale de la sursă, prin una sau mai multe porţi, până la destinaţie. Fiecare poartă menţine tabele spunând ce circuite virtuale o traversează, unde vor fi dirijate şi care este numărul noului circuit virtual.

Această schemă funcţionează cel mai bine atunci când toate reţelele au, în mare, aceleaşi proprietăţi. De exemplu, dacă toate garantează livrarea sigură a pachetelor de la nivelul reţea, atunci, exceptând un accident de-a lungul căii, fluxul de la sursă la destinaţie va fi de asemenea sigur. Similar, dacă nici una din ele nu garantează livrarea sigură, atunci concatenarea circuitelor virtuale nu este nici ea sigură. Pe de altă parte, dacă maşina sursă este într-o reţea care garantează livrarea sigură, dar una din reţelele intermediare poate pierde pachete, concatenarea schimbă fundamental natura serviciului.

Circuitele virtuale concatenate sunt, de asemenea, uzuale la nivelul transport. În particular, este posibil să se construiască o conductă de biţi folosind, să spunem, SNA, care se termină într-o poartă şi având o conexiune TCP de la această poartă la poarta următoare. În acest mod, un circuit virtual capăt-la-capăt poate fi construit acoperind diferite reţele şi protocoale.

5.5.4 Interconectarea reţelelor fără conexiuni

Modelul alternativ de interconectare este modelul datagramă, prezentat în fig. 5-46. În acest model, singurul serviciu pe care nivelul reţea îl oferă nivelului transport este capacitatea de a injecta datagrame în subreţea în speranţa că totul va merge bine. Nu există nici o noţiune de circuit virtual la nivelul reţea şi uitaţi de concatenarea lor. Acest model nu necesită ca toate pachetele care aparţin unei conexiuni să traverseze aceeaşi secvenţă de porţi. În fig. 5-46 sunt ilustrate datagramele de la gazda 1 pentru gazda 2, care urmează rute diferite prin reţeaua de interconectare. O decizie de dirijare este luată separat pentru fiecare pachet, eventual în funcţie de traficul din momentul în care este trimis pachetul. Această strategie poate utiliza rute multiple şi atinge astfel o capacitate mai mare decât modelul circuitelor virtuale concatenate. Pe de altă parte, nu există nici o garanţie că pachetele ajung la destinaţie în ordine, presupunând că vor ajunge.

Modelul din fig. 5-46 nu este aşa de simplu precum pare. Pe de o parte, dacă

fiecare reţea are propriul protocol de nivel reţea, nu este posibil ca un pachet dintr-o reţea să tranziteze alta. Se pot imagina rutere multiprotocol care încearcă să convertească dintr-un format în altul, dar, în afara cazului în care cele două formate sunt strâns înrudite având aceleaşi câmpuri de informaţie, aceste conversii vor fi incomplete şi deseori sortite eşecului. Din acest motiv, arareori se încearcă conversii.

O a doua problemă şi mai serioasă este adresarea. Să ne imaginăm un caz simplu: o gazdă din Internet încearcă să trimită un pachet IP către o gazdă dintr-o reţea adiacentă SNA. Adresele IP şi SNA sunt diferite. Ar trebui stabilită o corespondenţă între adresele IP şi SNA şi invers. În plus, ceea ce se poate adresa este diferit. În IP, gazdele (de fapt plăcile de interfaţă) au adrese. În SNA, pot avea adrese şi alte entităţi în afară de gazde (de exemplu echipamente hardware). În cel mai bun caz, cineva ar trebui să menţină o bază de date a tuturor corespondenţelor, dar ar fi o sursă constantă de necazuri.

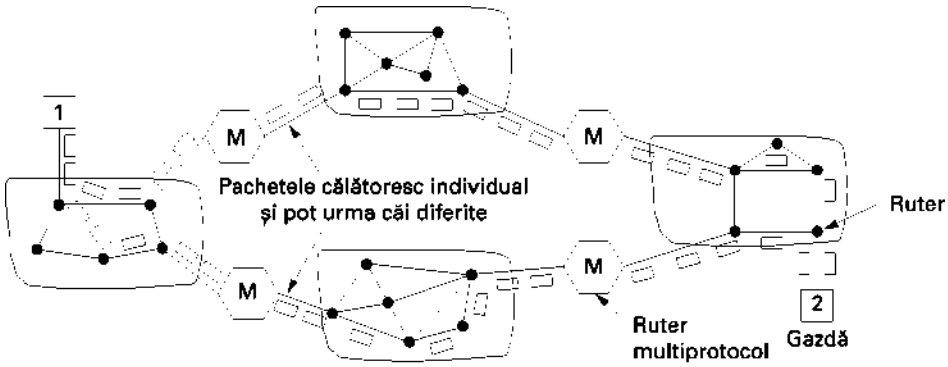


Fig. 5-46. O interconectare fără conexiuni.

O altă idee este de a proiecta un pachet universal „inter-reţea” şi a obliga toate ruterele să-l recunoască. Această abordare este, de fapt, chiar IP-ul – un pachet proiectat ca să fie purtat prin mai multe reţele. Bineînţeles se poate întâmpla ca IPv4 (protocolul actual din Internet) să scoată de pe piaţă toate celelalte protocoale, IPv6 (viitorul protocol Internet) să nu prindă şi să nu mai fie inventat nimic nou, dar istoria sugerează altceva. Este dificil ca toată lumea să fie de acord cu un singur format, atunci când companiile consideră că este în interesul lor să aibă formate proprii pe care să le controleze.

Să recapitulăm pe scurt cele două moduri prin care se poate aborda interconectarea reţelelor. Modelul circuitelor virtuale concatenate are, în esenţă, aceleaşi avantaje ca folosirea circuitelor virtuale într-o singură subreţea: zonele tampon pot fi rezervate în prealabil, poate fi garantată secvenţialita- tea, pot fi folosite antete scurte, iar necazurile cauzate de pachetele duplicate întârziate pot fi evitate.

El are, de asemenea, şi dezavantaje: spaţiul în tabele necesar în fiecare ruter pentru fiecare conexiune deschisă, lipsa unei dirijări alternative pentru a evita zonele congestionate şi vulnerabilitatea la defectarea ruterelor de pe parcurs. De asemenea, are dezavantajul de a fi dificil, dacă nu imposibil, de implementat în

cazul în care una din reţelele implicate este reţea nesigură de tip datagramă.

Proprietăţile abordării interconectării reţelelor prin datagrame sunt în mare parte aceleaşi ca şi cele ale subreţelelor de tip datagramă: potenţial de congestionare mai mare, dar de asemenea potenţial mai mare de adaptare la congestionări, robusteţe în cazul defectării ruterelor şi lungime mai mare necesară pentru antete. Într-o inter-reţea sunt posibili diferiţi algoritmi de dirijare adaptivă, aşa cum sunt în cadrul unei singure reţele de tip datagramă. Un avantaj major al abordării interconectării reţelelor prin datagrame este că aceasta poate fi folosită peste subreţele care nu folosesc circuite virtuale în interior. Multe LAN-uri, reţele mobile (de exemplu flotele aeriene şi navale) şi chiar unele WAN-uri intră în această categorie. Când o inter-reţea include una dintre acestea, apar probleme serioase dacă strategia de interconectare a reţelelor este bazată pe circuite virtuale.

5.5.5 Trecerea prin tunel

Rezolvarea cazului general de interconectare a două reţele diferite este extrem de dificilă. Cu toate acestea, există un caz special uzual care este gestionabil. Acest caz apare când gazdele sursă şi destinaţie sunt în acelaşi tip de reţea, dar între ele există o reţea diferită. Ca exemplu, gândiţi-vă la o bancă internaţională cu o reţea TCP/IP bazată pe Ethernet la Paris, o reţea TCP/IP bazată pe Ethernet la Londra şi o reţea non-IP de mare întindere (de exemplu ATM), aşa cum este prezentat în fig. 5-47.

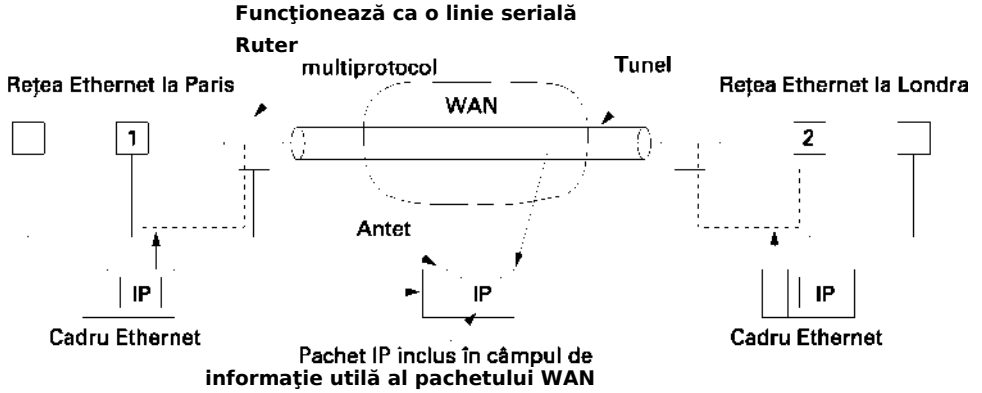


Fig. 5-47. Utilizarea tunelului pentru un pachet trimis de la Paris la Londra.

Soluţia acestei probleme este o tehnică numită **trecerea prin tunele**. Pentru a trimite un pachet IP la gazda 2, gazda 1 construieşte pachetul conţinând adresa IP a gazdei 2, îl inserează într-un cadru Ethernet adresat ruterului multiprotocol parizian şi apoi îl trimite în reţeaua Ethernet. Când ruterul multiprotocol primeşte cadrul, extrage pachetul IP, îl inserează în câmpul informaţie utilă al pachetului de nivel reţea WAN, pachet pe care îl adresează cu adresa ruterului multiprotocol londonez. Când ajunge acolo, ruterul londonez extrage pachetul IP şi îl trimite gazdei 2 în interiorul unui cadru Ethernet.

WAN-ul poate fi văzut ca un mare tunel ce se întinde de la un ruter multiprotocol la altul. Pachetul IP doar traversează tunelul de la un capăt la altul, aşezat confortabil în frumosul său lăcaş. El nu trebuie să aibă deloc grijă de comportarea la nivel WAN. Şi nici gazdele din oricare Ethernet. Numai ruterul multiprotocol trebuie să înţeleagă şi pachete IP şi WAN. Ca rezultat, întreaga distanţă de la mijlocul unui ruter multiprotocol până la mijlocul celuilalt acţionează ca o linie serială.

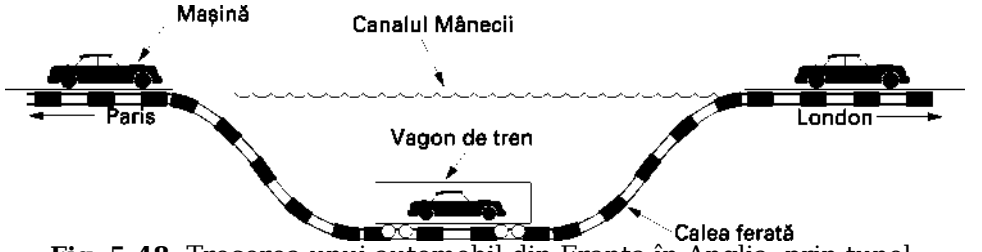


Fig. 5-48. Trecerea unui automobil din Franţa în Anglia, prin tunel.

O analogie poate face utilizarea tunelelor mai clară. Consideraţi o persoană conducând maşina de la Paris la Londra. În Franţa, maşina se deplasează sub acţiunea propriei puteri, dar când ajunge la Canalul Mânecii, este încărcată într-un tren de mare viteză şi transportată în Anglia prin Chunnel[§§§§§] (maşinile nu pot circula prin Chunnel). Efectiv, maşina este purtată ca informaţie utilă, aşa cum este prezentat în fig. 5-48. La capătul englez, maşina este eliberată pe drumurile engleze

§§§§§ N.T. Chunnel, ce provine de la Channel (canal) şi Tunnel (tunel) este denumirea dată în engleză tunelului de sub Canalul Mânecii.

şi continuă să se deplaseze, din nou cu propria putere. Utilizarea tunelelor printr-o reţea necunoscută funcţionează în acelaşi mod.

5.5.6 Dirijarea în reţele interconectate

Dirijarea printr-o reţea interconectată este similară cu dirijarea într-o singură subreţea, dar cu câteva complicaţii în plus. Să considerăm, de exemplu, interconectarea reţelelor din fig. 5-49(a) în care cinci reţele sunt conectate prin şase rutere (posibil multiprotocol). Crearea unui graf ca model al acestei situaţii este complicată de faptul că fiecare ruter poate accesa direct (mai clar, poate trimite pachete la) orice alt ruter conectat la orice reţea cu care este conectat. De exemplu, B din fig. 5- 49(a) poate accesa direct A şi C prin reţeaua 2 şi de asemenea D prin reţeaua 3. Aceasta duce la graful din fig. 5-49(b).

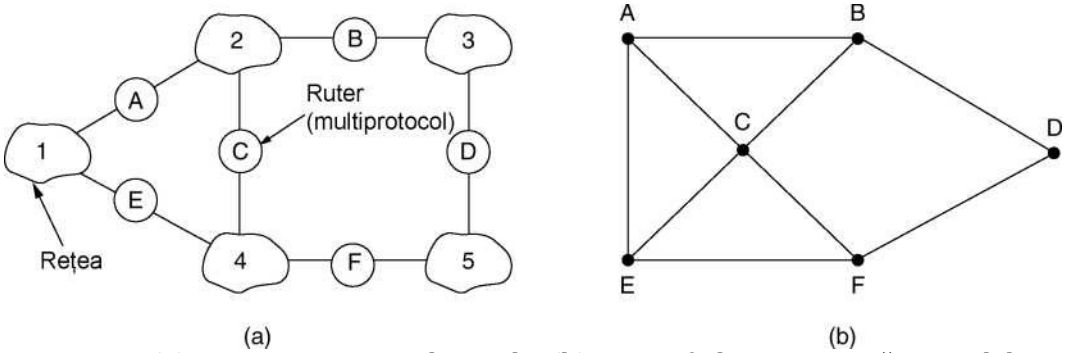


Fig. 5-49. (a) O interconectare de reţele. (b) Un graf al interconectării reţelelor.

O dată ce graful a fost construit, pe mulţimea de rutere multiprotocol pot fi aplicaţi algoritmii de dirijare cunoscuţi, cum ar fi algoritmii de tip vectori distanţă sau bazaţi pe starea legăturii. Aceasta duce la un algoritm de dirijare în doi paşi: în interiorul fiecărei reţele se foloseşte un **protocol de poartă interioară (internal gateway protocol)**, dar între reţele se foloseşte un **protocol de poartă exterioară (exterior gateway protocol)** („poartă” este un termen mai vechi pentru „ruter„). De fapt, din moment ce fiecare reţea este independentă, ele pot folosi algoritmi diferiţi. Deoarece fiecare reţea dintr-o interconectare de reţele este independentă de toate celelalte, este deseori referită ca un **sistem autonom (autonomous system-AS)**.

Un pachet tipic inter-reţele porneşte din LAN-ul său adresat ruterului multiprotocol local (în antetul nivelului MAC). După ce ajunge acolo, codul de la nivelul reţea decide cărui ruter multiprotocol să-i trimită pachetul, folosind propriile tabele de dirijare. Dacă la acel ruter se poate ajunge folosind protocolul de reţea nativ al pachetului, atunci este trimis direct acelui ruter. Altfel, este trimis utilizând tunele, încapsulat în protocolul cerut de reţeaua intermediară. Acest proces este repetat până când pachetul ajunge în reţeaua destinaţie.

Una din diferenţele dintre dirij area inter-reţele şi dirijarea intra-reţele este că dirijarea inter- reţele poate necesita deseori traversarea graniţelor internaţionale. Intră în joc legi diferite, cum ar fi legile suedeze despre secretul strict care se referă la exportarea din Suedia de date personale referitoare la cetăţenii suedezi. Un alt exemplu este legea canadiană care spune că traficul de date care porneşte

din Canada şi se termină în Canada nu poate părăsi ţara. Această lege înseamnă că traficul din Windsor, Ontario spre Vancouver nu poate fi dirijat prin apropiatul Detroit, chiar dacă această rută este cea mai rapidă şi cea mai ieftină.

O altă diferenţă între dirijarea internă şi cea externă este costul. Într-o singură reţea, se aplică în mod normal un singur algoritm de taxare. Cu toate acestea, diferite reţele pot avea administraţii diferite şi o cale poate fi mai ieftină decât alta. Similar, calitatea serviciului oferit de reţele diferite poate fi diferită şi acesta poate fi un motiv pentru alegerea unei căi în defavoarea alteia.

5.5.7 Fragmentarea

Fiecare reţea impune o anumită dimensiune maximă pentru pachetele sale. Aceste limite au diferite cauze, printre care:

1. Hardware (de exemplu, dimensiunea unui cadru Ethernet).
2. Sistemul de operare (de exemplu, toate zonele tampon au 512 octeţi).
3. Protocoale (de exemplu, numărul de biţi din câmpul lungime al pachetului).
4. Concordanţa cu unele standarde (inter)naţionale.
5. Dorinţa de a reduce la un anumit nivel retransmisiile provocate de erori.
6. Dorinţa de a preveni ocuparea îndelungată a canalului de către un singur pachet.

Rezultatul tuturor acestor factori este că proiectanţii de reţele nu au libertatea de a alege dimensiunea maximă a pachetelor oricum ar dori. Informaţia utilă maximă variază de la 8 octeţi (celulele ATM) la 65515 octeţi (pachetele IP), cu toate că dimensiunea pachetelor la nivelurile mai înalte este deseori mai mare.

O problemă evidentă apare când un pachet mare vrea să traverseze o reţea în care dimensiunea maximă a pachetului este prea mică. O soluţie este să ne asigurăm din capul locului că problema nu apare. Cu alte cuvinte, inter-reţeaua trebuie să utilizeze un algoritm de dirijare care evită transmiterea pachetelor prin reţele în care pachetele nu pot fi manevrate. Cu toate acestea, această soluţie nu este de fapt nici o soluţie. Ce se întâmplă dacă pachetul sursă original este prea mare pentru a fi manevrat de reţeaua destinaţie? Algoritmul de dirijare nu are cum să evite destinaţia.

În esenţă, singura soluţie a problemei este de a permite porţilor să spargă pachetele în **fragmente**, trimiţând fiecare pachet ca un pachet inter-reţea separat. Cu toate acestea, aşa cum ştie orice părinte al unui copil mic, convertirea unui obiect mare în fragmente mici este considerabil mai uşoară decât procesul invers. (Fizicienii au dat chiar un nume acestui efect: legea a doua a termodinamicii.) Reţelele cu comutare de pachete au, de asemenea, probleme în îmbinarea fragmentelor.

Există două strategii opuse pentru reconstituirea pachetului original din fragmente. Prima strategie este de a face fragmentarea cauzată de o reţea cu „pachete mici” transparentă pentru toate reţelele succesive prin care pachetul trebuie să treacă pe calea către destinaţia finală. Această opţiune este prezentată în fig. 5-50(a). În această strategie, reţeaua cu pachete mici are porţi (cel mai probabil, rutere specializate) către celelalte reţele. Când un pachet supradimensionat ajunge la poartă, poarta îl sparge în fragmente. Fiecare fragment este adresat

aceleiaşi porţi de ieşire, unde piesele sunt recombinate. În acest mod, trecerea printr-o reţea cu pachete mici a devenit transparentă. Reţelele următoare nici măcar nu sunt conştiente de fragmentarea făcută. Reţelele ATM, de exemplu, au hardware special pentru a oferi fragmentarea transparentă a pachetelor în celule şi apoi reasam- blarea celulelor în pachete. În lumea ATM, fragmentarea este numită segmentare; conceptul este acelaşi, dar diferă unele detalii.

Fragmentarea transparentă este simplă, dar are câteva probleme. Un motiv este că poarta de ieşire trebuie să ştie când a primit toate piesele, aşa încât în fiecare pachet trebuie inclus fie un câmp contor, fie un bit „sfârşit-de-pachet„. Un alt motiv este că toate pachetele trebuie să iasă prin aceeaşi poartă. Performanţele se pot degrada nepermiţând ca unele pachete să urmărească o cale către destinaţia finală şi alte pachete o cale diferită. O ultimă problemă este timpul suplimentar necesar pentru reasamblarea şi apoi refragmentarea repetată a unui pachet mare care traversează o serie de reţele cu pachete mici. ATM necesită fragmentare transparentă.

Cealaltă strategie de fragmentare este de a nu recombina fragmentele la nici o poartă intermediară. O dată ce un pachet a fost fragmentat, fiecare fragment este tratat ca şi cum ar fi un pachet original. Toate fragmentele sunt trecute printr-o poartă (sau porţi) de ieşire, aşa cum se arată în fig. 5-50(b). Recombinarea are loc doar la gazda destinaţie. Aşa funcţionează IP-ul.

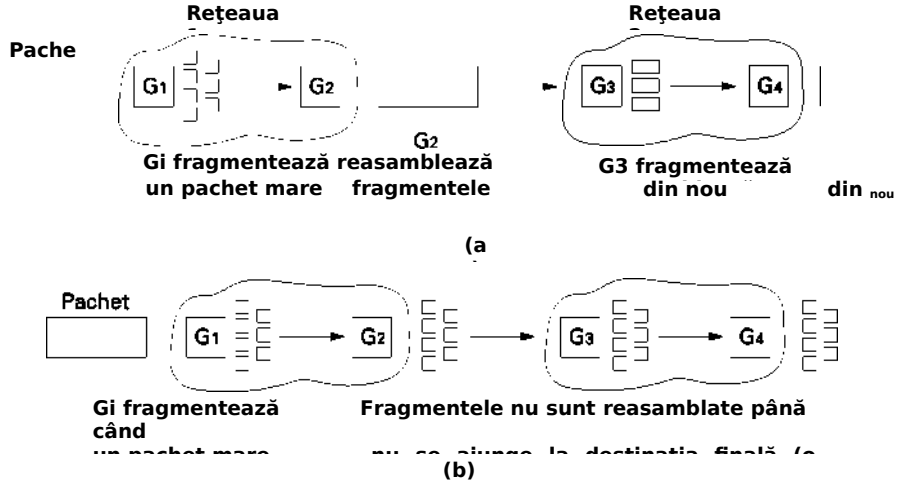


Fig. 5-50. (a) Fragmentare transparentă. (b) Fragmentare netransparentă.

Fragmentarea netransparentă are, de asemenea, unele probleme. De exemplu, necesită ca fiecare gazdă să fie capabilă să facă reasamblarea. Încă o problemă este că atunci când se fragmentează un pachet mare, supraîncărcarea creşte, deoarece fiecare fragment trebuie să aibă un antet. Pe câtă vreme în prima metodă această supraîncărcare dispare de îndată ce se iese din reţeaua cu pachete mici, în această metodă supraîncărcarea rămâne pentru restul călătoriei. Cu toate acestea, un avantaj al acestei metode este că se pot folosi mai multe porţi de ieşire şi se pot obţine performanţe mai bune. Desigur, dacă se foloseşte modelul circuitelor virtuale concatenate, acest avantaj nu este de nici un folos.

Când un pachet este fragmentat, fragmentele trebuie numerotate astfel încât şirul iniţial de date să poată fi reconstituit. O metodă de numerotare a fragmentelor este bazată pe un arbore. Dacă pachetul 0 trebuie descompus, bucăţile sunt numite 0.0, 0.1, 0.2 etc. Dacă aceste fragmente trebuie la rândul lor, să fie fragmentate mai târziu, bucăţile sunt numerotate 0.0.0, 0.0.1, 0.0.2, ..., 0.1.0, 0.1.1, 0.1.2 etc. Dacă în antet au fost rezervate suficiente câmpuri pentru cel mai rău caz şi nu sunt generate duplicate în altă parte, această schemă este suficientă pentru a asigura că

toate bucăţile pot fi corect reasamblate la destinaţie, neavând importanţă ordinea în care ajung.

Cu toate acestea, dacă o singură reţea pierde sau elimină pachete, apare necesitatea unor retransmisii capăt-la-capăt, cu efecte nefericite pentru schema de numerotare. Să presupunem că un pa

chet de 1024 biţi este iniţial fragmentat în patru fragmente de dimensiune egală, 0.0, 0.1, 0.2 şi 0.3. Fragmentul 0.1 este pierdut, dar toate celelalte părţi ajung la destinaţie. În cele din urmă, la sursă apare o depăşire de timp şi aceasta retransmite pachetul original. Numai că de această dată intervine legea lui Murphy şi calea trece printr-o reţea cu limita de 512 octeţi, deci sunt generate două fragmente. Când noul fragment 0.1 ajunge la destinaţie, receptorul va considera că toate cele patru bucăţi au ajuns şi va reconstrui pachetul incorect.

Un sistem de numerotare diferit (şi mai bun) este ca protocolul de interconectare a reţelelor să definească o dimensiune de fragment elementar suficient de mică ca fragmentul elementar să poată trece prin orice reţea. Când un pachet este fragmentat, toate bucăţile sunt egale cu dimensiunea fragmentului elementar, cu excepţia ultimului, care poate fi mai scurt. Un pachet inter-reţea poate conţine mai multe fragmente, din motive de eficienţă. Antetul inter-reţea trebuie să ofere numărul original al pachetului şi numărul fragmentului (sau primului fragment) elementar conţinut în pachet. Ca de obicei, trebuie să existe un bit care să indice că ultimul fragment elementar conţinut în pachetul inter-reţea este ultimul din pachetul original.

Această abordare necesită două câmpuri de secvenţă în antetul inter-reţea: numărul pachetului original şi numărul fragmentului. Există clar un compromis între dimensiunea fragmentului elementar şi numărul de biţi din numărul fragmentului. Deoarece dimensiunea fragmentului elementar este presupusă a fi acceptabilă pentru toate reţelele, fragmentările ulterioare ale unui pachet inter-reţea conţinând câteva fragmente nu cauzează probleme. În acest caz, limita extremă este reprezentată de un singur octet, numărul de fragment fiind reprezentat prin poziţia octetului în cadrul pachetului original, aşa cum se arată în fig. 5-51.

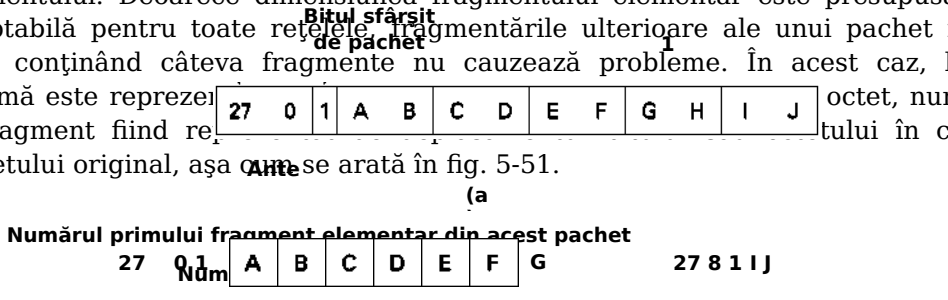


Fig. 5-51. Fragmentarea când dimensiunea datelor elementare este 1 octet.
(a) Pachetul original, conţinând 10 octeţi de date. (b) Fragmente după trecerea.

5.6 NIVELUL REŢEA ÎN INTERNET

Înainte de a intra în detaliile nivelului reţea din Internet merită studiate principiile care au ghidat proiectarea lui în trecut şi l-au făcut succesul care este astăzi. În ziua de azi, în prea multe cazuri oamenii par să le fi uitat. Aceste principii sunt enumerate şi discutate în RFC 1958, care merită citit (şi ar trebui să fie obligatoriu pentru toţi proiectanţii de protocoale – cu un examen final la sfârşit).

Acest RFC se bazează în mare parte pe idei care se găsesc în (Clark, 19 şi Saltzer, 1984). Vom rezuma ceea ce considerăm a fi cele 10 principii (de la cel mai important la cel mai puţin important).

1. **Fiţi siguri că funcţionează**. Nu finalizaţi proiectarea sau standardul până când mai multe prototipuri nu au comunicat cu succes unele cu altele. De prea multe ori proiectanţii întâi scriu un standard de 1000 de pagini, primesc aprobarea pentru el şi apoi descoperă că acest standard este eronat şi nu funcţionează. Apoi scriu versiunea 1.1 a standardului. Nu aşa se face.
2. **Menţine-l simplu**. Când există îndoieli, folosiţi soluţia mai simplă. William de Occam a enunţat acest principiu (briciul lui Occam) în secolul al 14-lea. Reformulat în termeni moderni: împotriviţi-vă caracteristicilor suplimentare. Dacă o caracteristică nu este absolut necesară, nu o includeţi, mai ales dacă acelaşi efect poate fi obţinut prin combinarea altor caracteristici.
3. **Faceţi alegeri clare**. Dacă există mai multe moduri de a realiza acelaşi lucru, alegeţi unul. A avea două sau mai multe moduri de a rezolva un lucru înseamnă să se caute necazul cu lumânarea. Standardele conţin de obicei multe opţiuni sau moduri sau parametri pentru că anumite grupări importante susţin că modul lor este cel mai bun. Proiectanţii ar trebui să reziste la această tendinţă. Spuneţi nu.
4. **Exploataţi modularitatea**. Acest principiu conduce direct la ideea de a avea stive de protocoale, fiecare nivel fiind independent de toate celelalte. În acest mod, dacă circumstanţele cer ca un modul sau nivel să fie schimbat, celelalte nu vor fi afectate.
5. **Aşteptaţi-vă la medii eterogene**. În orice reţea mare vor apărea diferite tipuri de hardware, posibilităţi de transmisie şi aplicaţii. Pentru a le trata pe toate, proiectarea reţelei trebuie să fie simplă, generală şi flexibilă.
6. **Evitaţi opţiuni sau parametri statici**. Dacă un parametru nu poate fi evitat (de exemplu dimensiunea maximă a unui pachet), este mai bine ca transmiţătorul şi receptorul să negocieze valoare decât să se definească valori fixe.
7. **Căutaţi o proiectare bună; nu este necesar să fie perfectă**. Deseori proiectanţii au un proiect bun care nu poate trata un caz mai ciudat. Decât să strice tot proiectul, proiectanţii ar trebui să-l păstreze şi să transfere povara rezolvării cazului celor cu cerinţe ciudate.
8. **Fiţi stricti atunci când trimiteţi şi toleranţi atunci când recepţionaţi**. Cu alte cuvinte, trimiteţi numai pachete care sunt în deplină conformitate cu standardul, dar aşteptaţi-vă ca pachetele recepţionate să nu fie în deplină conformitate cu standardul şi încercaţi să le folosiţi.
9. **Gândiţi-vă la scalabilitate**. Dacă sistemul trebuie să suporte milioane de gazde şi efectiv miliarde de utilizatori, nu se poate accepta nici un fel de bază de date centralizată, iar încărcarea trebuie distribuită cât mai uniform posibil folosind resursele disponibile.

10. **Luaţi în considerare performanţele şi costurile**. Dacă o reţea are performanţe slabe sau preţuri exorbitante, nu o va folosi nimeni.

Să lăsăm acum principiile generale şi să începem să studiem detaliile nivelului reţea din Internet. La nivelul reţea, Interne t-ul poate fi văzut ca o colecţie de subreţele sau **sisteme autonome** (eng.: **Autonomous Systems – AS**) care sunt interconectate. Nu există o structură reală, dar există câteva coloane vertebrale majore. Acestea sunt construite din linii de înaltă capacitate şi rutere rapide. La coloanele vertebrale sunt ataşate reţelele regionale (de nivel mediu), iar la aceste reţele regionale sunt ataşate LAN-urile din multe universităţi, companii şi furnizori de servicii Internet. O schiţă a acestei organizări cvasi-ierarhice este dată în fig. 5-52.

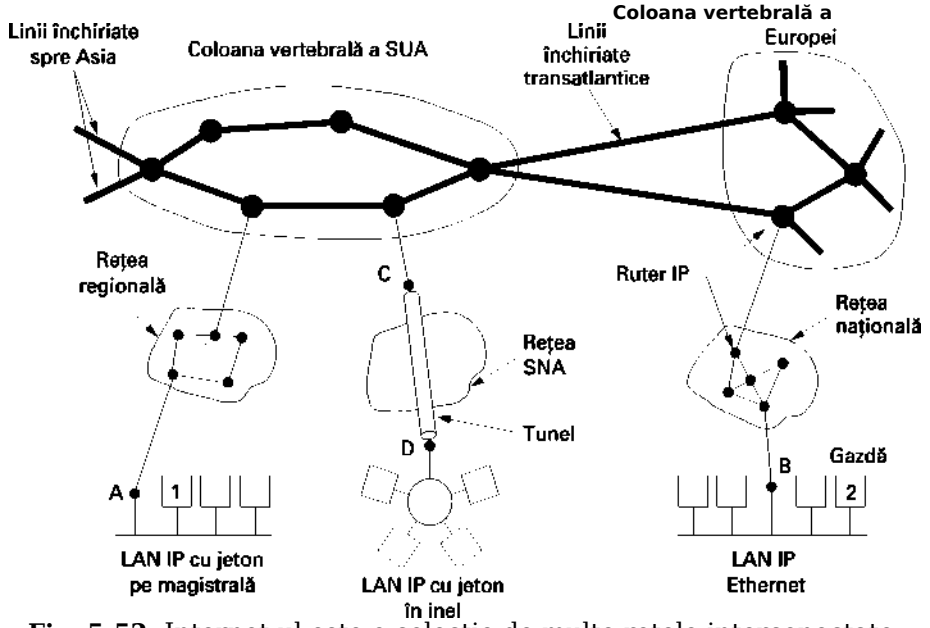


Fig. 5-52. Internet-ul este o colecţie de multe reţele interconectate.

Liantul care ţine Internet-ul la un loc este protocolul de nivel reţea, numit **IP** (**Internet Protocol –** protocolul Internet). Spre deosebire de protocoalele mai vechi de nivel reţea, acesta a fost proiectat de la început având în vedere interconectarea reţelelor. O metodă bună de a gândi nivelul reţea este aceasta. Sarcina lui este de a oferi cel mai bun mod posibil (adică negarantat) de a transporta datagrame de la sursă la destinaţie, fără a ţine seama dacă aceste maşini sunt sau nu în aceeaşi reţea sau dacă între ele există sau nu alte reţele.

Comunicaţia în Internet funcţionează după cum urmează. Nivelul transport preia şiruri de date şi le sparge în datagrame. În teorie, datagramele pot avea fiecare până la 64 KB, dar în practică, ele nu depăşesc 1500 octeţi (pentru a intra într-un cadru Ehternet) . Fiecare datagramă este transmisă prin Internet, fiind eventual fragmentată în unităţi mai mici pe drum. Când toate bucăţile ajung în sfârşit la maşina destinaţie, ele sunt reasamblate de nivelul reţea în datagrama originală.

Datagrama este apoi pasată nivelului transport, care o inserează în şirul de intrare al procesului receptor. După cum se poate vedea în fig. 5-52 un pachet transmis de gazda 1 trebuie să traverseze şase reţele pentru a ajunge la gazda 2. În practică se trece de obicei prin mult mai mult decât şase reţele.

5.6.1 Protocolul IP

Un loc potrivit pentru a porni studiul nostru despre nivelul reţea în Internet este însuşi formatul datagramelor IP. O datagramă IP constă dintr-o parte de antet şi o parte de text. Antetul are o parte fixă de 20 de octeţi şi o parte opţională cu lungime variabilă. Formatul antetului este prezentat în fig. 5-53. El este transmis în ordinea *big endian* (cel mai semnificativ primul): de la stânga la dreapta, începând cu bitul cel mai semnificativ al câmpului *Versiune.* (Procesorul SPARC este de tip *big endian;* Pentium este de tip *little endian* – cel mai puţin semnificativ primul). Pe maşinile de tip *little endian,* este necesară o conversie prin program atât la transmisie cât şi la recepţie.

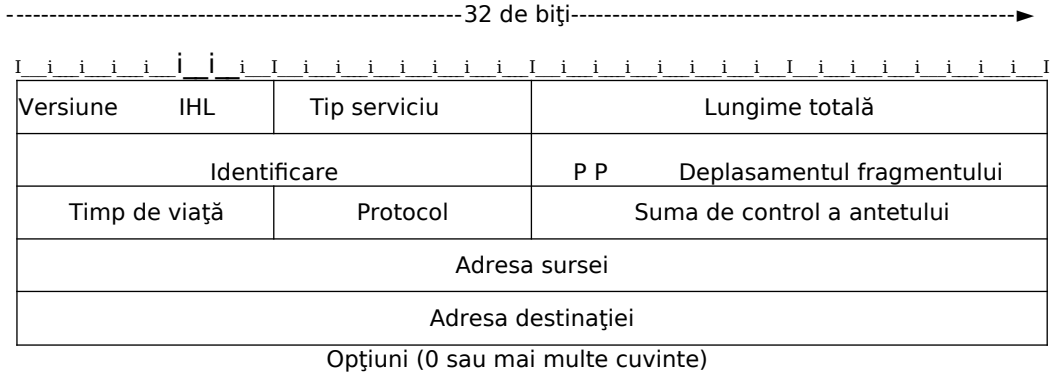


Fig. 5-53. Antetul IPv4 (protocolul Internet).

Câmpul *Versiune* memorează cărei versiuni de protocol îi aparţine datagrama. Prin includerea unui câmp versiune în fiecare datagramă, devine posibil ca tranziţia între versiuni să dureze ani de zile, cu unele maşini rulând vechea versiune, iar altele rulând-o pe cea nouă. La ora actuală are loc o tranziţie de la IPv4 la IPv6, care deja durează de câţiva ani şi în nici un caz nu s-a apropiat de final (Durand, 2001; Wilijakka, 2002; and Waddington and Chang 2002). Anumite persoane consideră chiar că nu se va întâmpla niciodată (Weiser, 2001). Referitor la numerotare, IPv5 a fost un protocol experimental de flux în timp real care nu a fost folosit pe scară largă.

Din moment ce lungimea antetului nu este constantă, un câmp din antet, *IHL,* este pus la dispoziţie pentru a spune cât de lung este antetul, în cuvinte de 32 de octeţi. Valoarea minimă este 5, care se aplică atunci când nu sunt prezente opţiuni. Valoarea maximă a acestui câmp de 4 biţi este 15, ceea ce limitează antetul la 60 de octeţi şi, astfel, câmpul de opţiuni la 40 de octeţi. Pentru unele opţiuni, cum ar fi cea care înregistrează calea pe care a mers un pachet, 40 de octeţi sunt mult prea puţini, făcând această opţiune nefolositoare.

Câmpul *Tip serviciu* este unul dintre puţinele câmpuri care şi-a schimbat sensul (oarecum) dea lungul anilor. A fost şi este în continuare menit să diferenţieze diferitele clase de servicii. Sunt posibile diferite combinaţii de fiabilitate şi viteză. Pentru vocea digitizată, livrarea rapidă are prioritate faţă de transmisia corectă. Pentru transferul de fişiere, transmisia fără erori este mai importantă decât transmisia rapidă.

La început, câmpul de 6 biţi conţinea (de la stânga la dreapta), un câmp *Precedenţă* de trei biţi şi trei indicatori, *D, T* şi *R.* Câmpul *Precedenţă* reprezintă prioritatea, de la 0 (normal) la 7 (pachet de control al reţelei). Cei trei biţi indicatori permiteau calculatorului gazdă să specifice ce îl afectează cel mai mult din mulţimea {Delay (Întârziere), Throughput (Productivitate), Reliability (Fiabilitate)}. În teorie, aceste câmpuri permit ruterelor să aleagă între, de exemplu, o legătură prin satelit cu o productivitate mare şi o întârziere mare sau o linie dedicată cu o productivitate scăzută şi o întârziere mică. În practică, ruterele curente ignoră adesea întregul câmp *Tip serviciu.*

Până la urmă, IETF a cedat şi a modificat puţin câmpul pentru a-l adapta la servicii diferenţiate. Şase dintre biţi sunt folosiţi pentru a indica căreia dintre clasele de servicii descrise mai devreme îi aparţine pachetul. Aceste clase includ patru priorităţi de introducere în coadă, trei probabilităţi de respingere şi clasele istorice.

Lungimea totală include totul din datagramă – atât antet cât şi date. Lungimea maximă este de 65535 octeţi. În prezent, această limită superioară este tolerabilă, dar în viitoarele reţele cu capacităţi de gigaocteţi vor fi necesare datagrame mai mari. Câmpul *Identificare* este necesar pentru a permite gazdei destinaţie să determine cărei datagrame îi aparţine un nou pachet primit. Toate fragmentele unei datagrame conţin aceeaşi valoare de *Identificare*.

Urmează un bit nefolosit şi apoi două câmpuri de 1 bit. *DF* însemnă Don't Fragment (A nu se fragmenta). Acesta este un ordin dat ruterelor ca să nu fragmenteze datagrama pentru că destinaţia nu este capabilă să reasambleze piesele la loc. De exemplu, când un calculator porneşte, memoria sa ROM poate cere să i se trimită o imagine de memorie ca o singură datagramă. Prin marcarea datagramei cu bitul *DF,* emiţătorul ştie că aceasta va ajunge într-o singură bucată, chiar dacă aceasta înseamnă că datagrama trebuie să evite o reţea cu pachete mai mici pe calea cea mai bună şi să aleagă o rută suboptimală. Toate maşinile trebuie să accepte fragmente de 576 octeţi sau mai mici.

MF însemnă More Fragments (mai urmează fragmente). Toate fragmentele, cu excepţia ultimului, au acest bit activat. El este necesar pentru a şti când au ajuns toate fragmentele unei datagrame.

Deplasamentul fragmentului spune unde este locul fragmentului curent în cadrul datagramei. Toate fragmentele dintr-o datagramă, cu excepţia ultimului, trebuie să fie un multiplu de 8 octeţi – unitatea de fragmentare elementară. Din moment ce sunt prevăzuţi 13 biţi, există un maxim de 8192 de fragmente pe datagramă, obţinându-se o lungime maximă a datagramei de 65536 octeţi, cu unul mai mult decât câmpul *Lungime totală.*

Câmpul *Timp de viaţă* este un contor folosit pentru a limita durata de viaţă a pachetelor. Este prevăzut să contorizeze timpul în secunde, permiţând un timp

maxim de viaţă de 255 secunde. El trebuie să fie decrementat la fiecare salt (hop – trecere dintr-o reţea în alta) şi se presupune că este decrementat de mai multe ori când stă la coadă un timp îndelungat într-un ruter. În practică, el con- torizează doar salturile. Când ajunge la valoarea zero, pachetul este eliminat şi se trimite înapoi la gazda sursă un pachet de avertisment. Această facilitate previne hoinăreala la infinit a datagramelor, ceea ce se poate întâmpla dacă tabelele de dirijare devin incoerente.

Când nivelul reţea a asamblat o datagramă completă, trebuie să ştie ce să facă cu ea. Câmpul *Protocol* spune cărui proces de transport trebuie să o predea. TCP este o posibilitate, dar tot aşa sunt şi UDP şi alte câteva. Numerotarea protocoalelor este globală la nivelul întregului Internet. Protocoalele şi alte numere alocate erau anterior definite în RFC 1700, dar astăzi ele sunt conţinute într-o bază de date on-line, care se găseşte la adresa *www.iana.org*.

Suma de control a antetului verifică numai antetul. O astfel de sumă de control este utilă pentru detectarea erorilor generate de locaţii de memorie proaste din interiorul unui ruter. Algoritmul este de a aduna toate jumătăţile de cuvinte, de 16 biţi, atunci când acestea sosesc, folosind aritmetică în complement faţă de unu şi păstrarea complementului faţă de unu al rezultatului. Pentru scopul acestui algoritm, se presupune că la sosire *suma de control a antetului* este zero. Acest algoritm este mai robust decât folosirea unei adunări normale. Observaţi că *suma de control a antetului* trebuie recalculată la fiecare salt, pentru că întotdeauna se schimbă cel puţin un câmp (câmpul *timp de viaţă),* dar se pot folosi trucuri pentru a accelera calculul.

Adresa sursei şi *Adresa destinaţiei* indică numărul de reţea şi numărul de gazdă. Vom discuta adresele Internet în secţiunea următoare. Câmpul *Opţiuni* a fost proiectat pentru a oferi un subterfugiu care să permită versiunilor viitoare ale protocolului să includă informaţii care nu sunt prezente în proiectul original, pentru a permite cercetătorilor să încerce noi idei şi pentru a evita alocarea unor biţi din antet pentru informaţii folosite rar. Opţiunile sunt de lungime variabilă. Fiecare începe cu un cod de un octet care identifică opţiunea. Unele opţiuni sunt urmate de un câmp de un octet reprezentând lungimea opţiunii, urmat de unul sau mai mulţi octeţi de date. Câmpul *Opţiuni* este completat până la un multiplu de 4 octeţi. Iniţial erau definite cinci opţiuni, aşa cum sunt listate în fig. 5-54, dar de atunci au mai fost adăugate şi altele. Lista completă se găseşte la adresa *www.iana.org/assignementslip-parameters*.

Opţiune	**Descriere**
Securitate	Menţionează cât de secretă este datagrama
Dirijare strictă de la sursă	Indică calea completă de parcurs
Dirijare aproximativă de la sursă	Indică o listă a ruterelor care nu trebuie sărite
Înregistrează calea	Determină fiecare ruter să-şi adauge adresa IP
Amprentă de timp	Determină fiecare ruter să-şi adauge adresa şi o amprentă

Fig. 5-54. Unele dintre opţiunile IP.

Opţiunea *Securitate* menţionează cât de secretă este informaţia. În teorie, un ruter militar poate folosi acest câmp pentru a menţiona că nu se doreşte o dirijare prin anumite ţări pe care militarii le consideră a fi „băieţii răi”. În practică, toate ruterele îl ignoră, deci singura sa funcţie practică este să ajute spionii să găsească

mai uşor lucrurile de calitate.

Opţiunea *Dirijare strictă de la sursă* dă calea completă de la sursă la destinaţie ca o secvenţă de adrese IP. Datagrama este obligată să urmărească această cale precisă. Ea este deosebit de utilă pentru administratorii de sistem pentru a trimite pachete de urgenţă atunci când tabelele de dirijare sunt distruse sau pentru a realiza măsurători de timp.

Opţiunea *Dirijare aproximativă de la sursă* cere ca pachetul să traverseze o listă specificată de rutere şi în ordinea specificată, dar este permisă trecerea prin alte rutere pe drum. În mod normal, această opţiune ar putea oferi doar câteva rutere, pentru a forţa o anumită cale. De exemplu, pentru a forţa un pachet de la Londra la Sydney să meargă spre vest în loc de est, această opţiune poate specifica rutere în New York, Los Angeles şi Honolulu. Această opţiune este foarte utilă atunci când motive politice sau economice dictează trecerea prin anumite ţări sau evitarea lor.

Opţiunea *Înregistrează calea* indică ruterelor de pe cale să-şi adauge adresele IP la câmpul opţiune. Aceasta permite administratorilor de sistem să localizeze pene în algoritmii de dirijare („De ce pachetele de la Houston la Dallas trec mai întâi prin Tokio?”). Când reţeaua ARPANET a fost înfiinţată, nici un pachet nu trecea vreodată prin mai mult de nouă rutere, deci 40 de octeţi pentru opţiuni au fost destui. Aşa cum s-a menţionat anterior, acum dimensiunea este prea mică .

În sfârşit, opţiunea *Amprentă de timp* este similară opţiunii *Înregistrează ruta,* cu excepţia faptului că, în plus faţă de înregistrarea adresei sale de 32 de biţi, fiecare ruter înregistrează şi o amprentă de timp de 32 de biţi. Şi această opţiune este folosită în special pentru depanarea algoritmilor de dirijare.

5.6.2 Adrese IP

Fiecare gazdă şi ruter din Internet are o adresă IP, care codifică adresa sa de reţea şi de gazdă. Combinaţia este unică: în principiu nu există două maşini cu aceeaşi adresă IP. Toate adresele IP sunt de 32 de biţi lungime şi sunt folosite în câmpurile *Adresă sursă* şi *Adresă destinaţie* ale pachetelor IP. Este important de observat că o adresă IP nu se referă de fapt la o gazdă. Se referă de fapt la o interfaţă de reţea, deci dacă o gazdă este în două reţele, trebuie să folosească două adrese IP. Totuşi în practică, cele mai multe gazde sunt conectate la o singură reţea şi deci au o adresă IP.

Timp de mai multe decenii, adresele IP erau împărţite în cinci categorii ilustrate în fig. 5-55. Acest model de alocare a fost denumit **clase de adrese**. Nu mai este folosit, dar referinţele la acest model sunt în continuare des întâlnite în literatură. Vom discuta puţin mai târziu ce a înlocuit mode- Iul claselor de adrese.

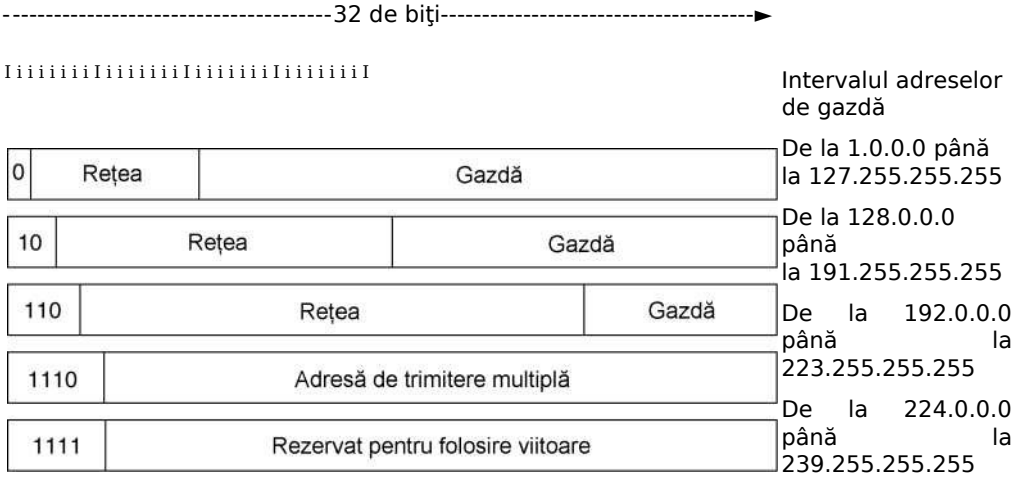

32 de biţi
I i i i i i i i I i i i i i i i I i i i i i i i I i i i i i i i I
Clasă
Intervalul adreselor de gazdă
0
Rețea
Gazdă
De la 1.0.0.0 până la 127.255.255.255
10
Rețea
Gazdă
De la 128.0.0.0 până la 191.255.255.255
110
Rețea
Gazdă
De la 192.0.0.0 până la 223.255.255.255
1110
Adresă de trimitere multiplă
De la 224.0.0.0 până la 239.255.255.255
1111
Rezervat pentru folosire viitoare
De la 240.0.0.0 până la 255.255.255.255

Fig. 5-55. Formatul

Formatele de clasă A, B, C şi D permit până la 128 reţele cu 16 milioane de

gazde fiecare, 16.384 reţele cu până la 64K gazde, 2 milioane de reţele (de exemplu, LAN-uri) cu până la 256 gazde fiecare (deşi unele dintre acestea sunt speciale). De asemenea este suportată şi trimiterea multiplă (multicast), în care fiecare datagramă este direcţionată mai multor gazde. Adresele care încep cu 1111 sunt rezervate pentru o folosire ulterioară. Peste 500 000 de reţele sunt conectate acum la Internet şi numărul acestora creşte în fiecare an. Pentru a evita conflictele numerele de reţea sunt atribuite de **ICANN (Internet Corporation for Assigned NAMES and Numbers –** Corporaţia Internet

pentru numere şi nume atribuite). La rândul său, ICANN a împuternicit diverse autorităţi regionale să administreze părţi din spaţiul de adrese şi acestea, la rândul lor, au împărţit adrese ISP-urilor şi altor companii.

Adresele de reţea, care sunt numere de 32 de biţi, sunt scrise în mod uzual în **notatia zecimală cu punct**. În acest format, fiecare din cei 4 octeţi este scris în zecimal, de la 0 la 255. De exemplu, adresa hexazecimală C0290614 este scrisă ca 192.41.6.20. Cea mai mică adresă IP este 0.0.0.0 şi cea mai mare este 255.255.255.255.

Valorile 0 şi -l au semnificaţii speciale, aşa cum se arată în fig. 5-56. Valoarea 0 înseamnă reţeaua curentă sau gazda curentă. Valoarea -l este folosită ca o adresă de difuzare pentru a desemna toate gazdele din reţeaua indicată.

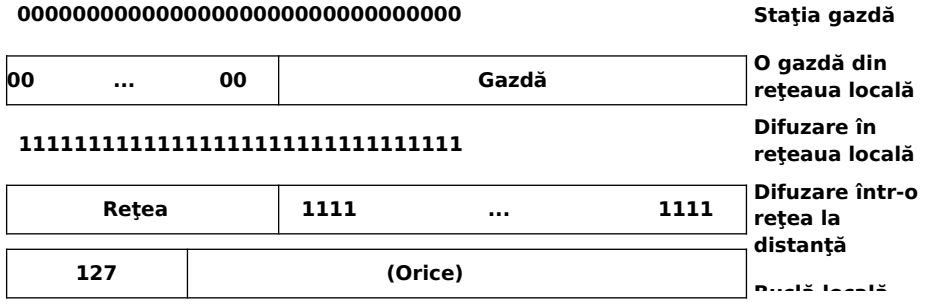


Fig. 5-56. Adrese IP

Adresa IP 0.0.0.0 este folosită de gazde atunci când sunt pornite. Adresele IP cu 0 ca număr de reţea se referă la reţeaua curentă. Aceste adrese permit ca maşinile să refere propria reţea fără a cunoaşte numărul de reţea (dar ele trebuie să cunoască clasa adresei pentru a şti câte zerouri să includă). Adresele care constau numai din l-uri permit difuzarea în reţeaua curentă, în mod uzual un LAN. Adresele cu un număr exact de reţea şi numai l-uri în câmpul gazdă permit maşinilor să trimită pachete de difuzare în LAN-uri la distanţă, aflate oriunde în Internet (deşi mulţi administratori de sistem dezactivează această opţiune). În final, toate adresele de forma 127.xx.yy.zz sunt rezervate pentru testări în buclă locală (loopback). Pachetele trimise către această adresă nu sunt trimise prin cablu; ele sunt prelucrate local şi tratate ca pachete sosite. Aceasta permite trimiterea pachetelor către reţeaua locală fără ca emiţătorul să-i cunoască numărul.

Subretele

Aşa cum am văzut, toate gazdele dintr-o reţea trebuie să aibă acelaşi număr de reţea. Această proprietate a adresării IP poate crea probleme când reţeaua creşte. De exemplu, să considerăm o universitate care a folosit la început o reţea de clasă B pentru calculatoarele din reţeaua Ethernet a Departamentului de Calculatoare. Un an mai târziu, departamentul de Inginerie Electrică a vrut acces la Internet, deci a cumpărat un repetor pentru a extinde Ethernetul Departamentului de Calculatoare în clădirea lor. Cu timpul, multe alte departamente au achiziţionat calculatoare şi limita de patru repetoare per Ethernet a fost rapid atinsă. Era nevoie de o altă organizare.

Achiziţionarea altei adrese de reţea ar fi fost dificilă deoarece adresele sunt

insuficiente şi universitatea avea deja adrese suficiente pentru peste 60,000 de staţii. Problema provine de la regula care afirmă că o singură adresă de clasă A, B sau C se referă la o singură reţea, nu la o mul-

ţime de reţele. Cum tot mai multe organizaţii s-au lovit de această problemă, sistemul de adresare a fost puţin modificat pentru a o rezolva

Soluţia este să se permită ca o reţea să fie divizată în mai multe părţi pentru uz intern, dar pentru lumea exterioară să se comporte tot ca o singură reţea. În ziua de azi o reţea de campus tipică poate arăta ca în fig. 5-57, cu un ruter principal conectat la un ISP sau la reţeaua regională şi numeroase reţele Ethernet împrăştiate prin campus în diferite departamente. Fiecare Ehternet are propriul ruter conectat la ruterul principal (posibil printr-un LAN coloană vertebrală, dar natura conexiunii interruter nu este relevantă aici)

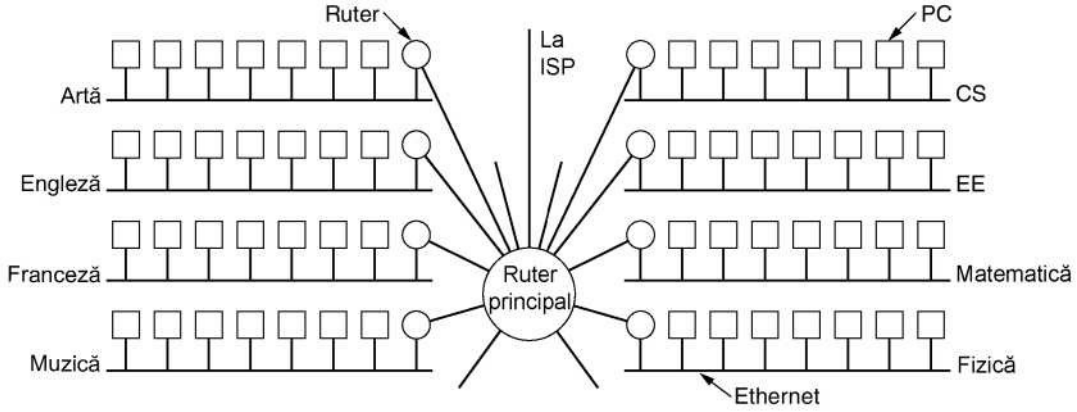


Fig. 5-57. O reţea de campus compusă din mai multe LAN-uri ale diferitelor departamente

În literatura Internet, aceste părţi sunt numite **subretele**. Aşa cum am menţionat în Cap. 1, această utilizare intră în conflict cu termenul „subreţea”, care înseamnă mulţimea tuturor ruterelor şi liniilor de comunicaţie dintr-o reţea. Din fericire, va fi clar din context care semnificaţie este atribuită. În această secţiune, şi în următoarea definiţia folosită va fi cea nouă.

Atunci când un pachet ajunge la ruterul principal, de unde ştie acesta către ce subreţea (Ethernet) să-l trimită? O soluţie ar fi să existe o tabelă cu 65,536 înregistrări în ruterul principal care să-i spună acestuia ce ruter să folosească pentru fiecare staţie din campus. Această idee ar funcţiona, dar ar fi nevoie de o tabelă foarte mare în ruterul principal şi de multă muncă manuală de întreţinere pe măsură ce staţiile ar fi adăugate, mutate sau scoase din uz.

În locul ei a fost inventată o altă soluţie. În esenţă, în loc să existe o singură adresă de clasă B, cu 14 biţi pentru numărul reţelei şi 16 biţi pentru gazdă, un număr de biţi din numărul gazdei sunt folosiţi pentru a crea un număr de subreţea. De exemplu, dacă o universitate are 35 de departamente, ar putea folosi un număr de subreţea de 6 biţi şi un număr de 10 biţi pentru gazde, ceea ce ar permite un număr de 64 de reţele Ethernet, fiecare cu un maxim de 1022 de gazde (aşa cum am menţionat mai devreme, 0 şi -l nu sunt disponibile). Această împărţire ar putea fi modificată ulterior în caz că a fost o greşeală.

Pentru a se putea folosi subreţele, ruterul principal trebuie are nevoie de o

mască de subreţea, care indică separarea dintre numărul reţea + subreţea şi gazdă, aşa cum este ilustrat în fig. 5-58. Măştile de subreţea sunt scrise de asemenea în notaţie zecimală cu punct, cu adăugarea unui caracter / (slash) urmat de numărul de biţi din partea cu reţea + subreţea. Pentru exemplul din fig. 5-58, masca de subreţea poate fi scrisă ca 255.255.255.0 . O notaţie alternativă este /22 pentru a indica că masca subreţelei are lungimea de 22 de biţi.

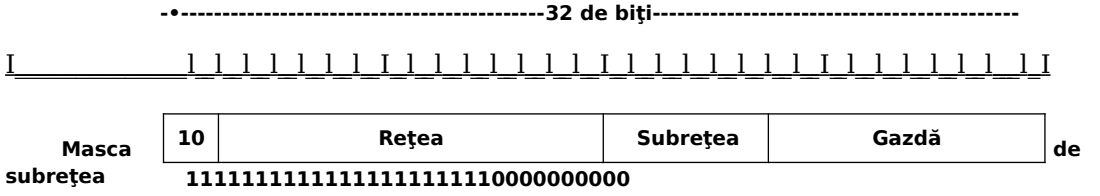


Fig. 5-58. O reţea de clasă B împărţită în 64 de subreţele

În afara reţelei, împărţirea în subreţele nu este vizibilă, astfel încât alocarea unei noi subreţele nu necesită contactarea ICANN sau schimbarea unor baze de date externe. În acest exemplu, prima subreţea poate folosi adrese IP începând de la 130.50.4.1, cea de-a doua poate începe de la 130.50.8.1, cea de-a treia poate începe de la 130.50.12.1 şi aşa mai departe. Pentru a se înţelege de ce numărul subreţelei creşte cu patru, să observăm că adresele binare corespunzătoare sunt următoarele:

Subreţea	10000010	00110010	000001|00	000000
Subreţea	10000010	00110010	000010|00	000000
Subreţea	10000010	00110010	000011|00	000000

Aici, bara verticală (|) arată graniţa dintre numărul subreţelei şi numărul gazdei. La stânga se găseşte numărul de 6 biţi al subreţelei, la dreapta numărul de 10 biţi al gazdei.

Pentru a vedea cum funcţionează subreţelele, este necesar să explicăm cum sunt prelucrate pachetele IP într-un ruter. Fiecare ruter are o tabelă ce memorează un număr de adrese IP de forma (reţea, 0) şi un număr de adrese IP de forma (această-reţea, gazdă). Primul tip indică cum se ajunge la reţelele aflate la distanţă. Al doilea tip spune cum se ajunge la gazdele locale. Fiecărei tabele îi este asociată interfaţa de reţea care se foloseşte pentru a ajunge la destinaţie şi alte câteva informaţii.

Când soseşte un pachet IP, adresa destinaţie este căutată în tabela de dirijare. Dacă pachetul este pentru o reţea aflată la distanţă, el este trimis ruterului următor prin interfaţa specificată în tabelă. Dacă este o gazdă locală (de exemplu în LAN-ul ruterului), pachetul este trimis direct către destinaţie. Dacă reţeaua nu este prezentă, pachetul este trimis unui ruter implicit care are tabele mai extinse. Acest algoritm înseamnă că fiecare ruter trebuie să memoreze numai reţele şi gazde, nu perechi (reţea, gazdă), reducând considerabil dimensiunea tabelelor de dirijare.

Când este introdusă împărţirea în subreţele, tabelele de dirijare sunt schimbate, adăugând intrări de forma (această-reţea, subreţea, 0) şi (această-reţea, această-subreţea, gazdă). Astfel un ruter din subreţeaua *k* ştie cum să ajungă la toate celelalte subreţele şi, de asemenea, cum să ajungă la toate gazdele din subreţeaua *k.* El nu trebuie să cunoască detalii referitoare la gazde din alte subreţele. De fapt, tot ceea ce trebuie schimbat este de a impune fiecărui ruter să facă un ŞI logic cu **masca de subreţea** a reţelei pentru a scăpa de numărul de gazdă şi a căuta adresa rezultată în tabelele sale (după ce determină cărei clase de reţele aparţine). De exemplu, asupra unui pachet adresat către
130.50.15.6 care ajunge la ruterul principal se face un ŞI logic cu masca de subreţea 255.255.252.0/22 pentru a obţine adresa 130.50.12.0. Această adresă este căutată în tabelele de dirijare pentru a se găsi cum se ajunge la gazdele din

subreţeaua 3. Ruterul din subreţeaua 5 este astfel uşurat de munca de a memora toate adresele de nivel legătură de date ale altor gazde decât cele din subreţeaua 5. Împărţirea în subreţele reduce astfel spaţiul tabelelor de dirijare prin crearea unei ierarhii pe trei niveluri, alcătuită din reţea, subreţea şi gazdă.

CIDR – Dirijarea fără clase între domenii

IP este folosit intens de câteva decenii. A funcţionat extrem de bine, aşa cum a fost demonstrat de creşterea exponenţială a Internet-ului. Din nefericire, IP devine rapid o victimă a propriei popularităţi: îşi epuizează adresele. Acest dezastru fantomatic a generat foarte multe discuţii şi controverse în cadrul comunităţii Internet referitor la cum să fie tratat. În această secţiune vom descrie atât problema cât şi câteva soluţii propuse.

Prin 1987, câţiva vizionari au prezis că într-o zi Internet-ul poate creşte până la 100.000 de reţele. Cei mai mulţi experţi s-au exprimat cu dispreţ că aceasta va fi peste decenii, dacă va fi vreodată. Cea de-a 100.000-a reţea a fost conectată în 1996. Problema, expusă anterior, este că Internet-ul îşi epuizează rapid adresele IP. În principiu, există peste 2 miliarde de adrese, dar practica organizării spaţiului de adrese în clase (vezi fig. 5-55) iroseşte milioane din acestea. În particular, adevăraţii vinovaţi sunt de reţelele de clasă B. Pentru cele mai multe organizaţii, o adresă de clasă A, cu 16 milioane de adrese este prea mare, iar o reţea de clasă C, cu 256 de adrese, este prea mică. O reţea de clasă B, cu 65.536 adrese, este numai bună. În folclorul Internet, această situaţie este cunoscută ca **problema celor trei urşi** (ca din *Goldilocks and the Three Bears).*

În realitate, o adresă de clasă B este mult prea mare pentru cele mai multe organizaţii. Studiile au arătat că mai mult de jumătate din toate reţelele de clasă B au mai puţin de 50 de gazde. O reţea de clasă C ar fi fost suficientă, dar fără îndoială că fiecare organizaţie care a cerut o adresă de clasă B a crezut că într-o zi ar putea depăşi câmpul gazdă de 8 biţi. Privind retrospectiv, ar fi fost mai bine să fi avut reţele de clasă C care să folosească 10 biţi în loc de opt pentru numărul de gazdă, permiţând 1022 gazde pentru o reţea. În acest caz, cele mai multe organizaţii s-ar fi decis, probabil, pentru o reţea de clasă C şi ar fi existat jumătate de milion dintre acestea (comparativ cu doar 16.384 reţele de clasă B).

Este greu să fie învinuiţi proiectanţii Internetului pentru că nu au pus la dispoziţie mai multe adrese de clasă B (şi mai mici). În momentul în care s-a luat decizia să fie create cele trei clase, Internetul era o reţea de cercetare care conecta marile universităţi din SUA (plus un număr mic de companii şi sit-uri militare care făceau cercetări în domeniul reţelelor). Pe atunci nimeni nu a perceput Internetul ca fiind un sistem de comunicaţie în masă care va rivaliza cu reţeaua telefonică. Fără îndoială că în acel moment cineva a spus: „SUA are aproximativ 2000 de universităţi şi colegii. Chiar dacă toate se conectează la Internet şi se mai conectează şi multe universităţi din alte ţări, nu o să ajungem niciodată la 16.000 pentru că nu există atâtea universităţi în întreaga lume. În plus faptul că numărul gazdei este un număr întreg de octeţi, măreşte viteza de prelucrare a pachetelor.”

Totuşi, dacă s-ar fi alocat 20 de biţi numărului de reţea pentru reţele de clasă B, ar fi apărut o altă problemă: explozia tabelelor de dirijare. Din punctul de vedere al ruterelor, spaţiul de adrese IP este o ierarhie pe două niveluri, cu numere de reţea

şi numere de gazde. Ruterele nu trebuie să ştie despre toate gazdele, dar ele trebuie să ştie despre toate reţelele. Dacă ar fi fost în folosinţă jumătate de milion de reţele de clasă C, fiecare ruter din întregul Internet ar fi necesitat o tabelă cu o jumătate de milion de intrări, una pentru fiecare reţea, spunând care linie se foloseşte pentru a ajunge la respectiva reţea, împreună cu alte informaţii.

Memorarea fizică efectivă a tabelelor cu jumătate de milion de intrări este probabil realizabilă, deşi costisitoare pentru ruterele critice care trebuie să menţină tabelele în RAM static pe plăcile I/O. O problemă mai serioasă este reprezentată de creşterea complexităţii diferiţilor algoritmi referitori la gestiunea tabelelor, care este mai rapidă decât liniară. Şi mai rău, o mare parte din programele şi firmware-ul ruterelor existente a fost proiectat pe vremea când Internet-ul avea 1000 de reţele conectate şi 10.000 de reţele păreau la depărtare de decenii. Deciziile de proiectare făcute atunci sunt departe de a fi optime acum.

În plus, diferiţi algoritmi de dirijare necesită ca fiecare ruter să-şi transmită tabelele periodic (de exemplu protocolul vectorilor distanţă). Cu cât tabelele sunt mai mari, cu atât este mai probabil ca anumite părţi să se piardă pe drum, ducând la date incomplete la celălalt capăt şi, posibil, la instabilităţi de dirijare.

Problema tabelelor de dirijare ar fi putut fi rezolvată prin adoptarea unei ierarhii mai adânci. De exemplu, ar fi mers dacă s-ar fi impus ca fiecare adresă să conţină un câmp ţară, judeţ/provincie, oraş, reţea şi gazdă. Atunci fiecare ruter ar fi trebuit să ştie doar cum să ajungă la fiecare ţară, la judeţele sau provinciile din ţara sa, oraşele din propriul judeţ sau provincie şi reţelele din oraşul său. Din nefericire, această soluţie ar necesita mai mult de 32 de biţi pentru adrese IP şi ar folosi adresele ineficient (Liechtenstein ar fi avut tot atâţia biţi ca Statele Unite).

Pe scurt, cele mai multe soluţii rezolvă o problemă, dar creează o alta. Soluţia care a fost implementată acum şi care a dat Internet-ului un pic de spaţiu de manevră este **CIDR (Classless InterDomain Routing –** Dirijarea fără clase între domenii). Ideea de la baza CIDR, descrisă în RFC 1519, este de a aloca adresele IP rămase, în blocuri de dimensiune variabilă, fără a se ţine cont de clase. Dacă un sit are nevoie, să zicem, de 2000 de adrese, îi este dat un bloc de 2048 adrese la o graniţă de 2048 de octeţi.

Renunţarea la clase face rutarea mai complicată. În vechiul sistem cu clase, rutarea se efectua în felul următor. Atunci când un pachet ajungea la un ruter, o copie a adresei IP era deplasată la dreapta cu 28 de biţi pentru a obţine un număr de clasă de 4 biţi. O ramificaţie cu 16 căi sorta pachetele în A, B, C şi D (dacă era suportat), cu 8 cazuri pentru clasa A, patru cazuri pentru clasa B, două cazuri pentru clasa C şi câte unul pentru D şi E. Programul pentru fiecare clasă masca numărul de reţea de 8, 16 sau 24 de biţi şi îl alinia la dreapta într-un cuvânt de 32 de biţi. Numărul de reţea era apoi căutat în tabela pentru A, B sau C, de obicei indexat pentru reţelele A şi B şi folosind dispersia (hashing) pentru reţelele C. Odată intrarea găsită, se putea găsi linia de ieşire şi pachetul era retransmis.

Cu CIDR, acest algoritm simplu nu mai funcţionează. În loc de aceasta, fiecare intrare din tabela de rutare este extinsă cu o mască de 32 de biţi. Din acest motiv, acum există o singură tabelă de rutare pentru toate reţelele, constituită dintr-un un vector de triplete (adresă IP, mască subreţea, linie de ieşire). Când soseşte un pachet IP, mai întâi se extrage adresa IP a destinaţiei. Apoi (conceptual) tabela de

rutare este scanată intrare cu intrare, mascând adresa destinaţie şi comparând cu intrarea din tabelă, în căutarea unei potriviri. Este posibil ca mai multe intrări (cu măşti de subreţea de lungimi diferite) să se potrivească, caz în care este folosită cea mai lungă mască. Astfel, dacă există potrivire pentru o mască /20 şi pentru o mască /24, este folosită intrarea cu /24.

Pentru a accelera procesul de găsire a adreselor au fost imaginaţi algoritmi complecşi (Ruiz- Sanchez et al., 2001). Ruterele comerciale folosesc chip-uri VLSI proprietare cu aceşti algoritmi implementaţi în hardware.

Pentru a face algoritmul de retransmitere mai uşor de înţeles, să considerăm un exemplu în care sunt disponibile milioane de adrese, începând de la 194.24.0.0. Să presupunem că Universitatea Cambridge are nevoie de 2048 de adrese şi are atribuite adresele de la 194.24.0.0 până la
194.24.7.255, împreună cu masca 255.255.248.0. Apoi, Universitatea Oxford cere 4096 de adrese. Deoarece un bloc de 4096 de adrese trebuie să fie aliniat la o frontieră de 4096 octeţi, acestea nu pot fi numerotate începând de la 194.8.0.0. În loc, se primesc adrese de la 194.24.16.0 până la
194.24.31.255, împreună cu o mască de 255.255.240.0. Acum Universitatea din Edinburgh cere

1024 de adrese şi îi sunt atribuite adresele de la 194.24.8.0 până la 194.24.11.255 şi masca 255.255.252.0. Alocările sunt rezumate în fig. 5-59.

Universitatea	Prima adresă	Ultima adresă	Număr adrese	Notaţie
Cambridge	194.24.0.0	194.24.7.255	2048	194.24.0.0/21
Edinburgh	194.24.8.0	194.24.11.255	1024	194.24.8.0/22
(disponibil)	194.24.12.0	194.24.15.255	1024	194.24.12/22
Oxford	194.24.16.0	194.24.31.255	4096	194.24.16.0/20

Fig. 5-59. Un set de atribuiri de adrese IP.

Tabelele de dirijare din toată lumea sunt acum actualizate cu cele trei intrări atribuite. Fiecare intrare conţine o adresă de bază şi o mască de subreţea. Aceste intrări (în binar) sunt:

	Adresă	Mască
C	11000010 00011000 00000000 00000000	11111111 11111111 11111000 00000000
E	11000010 00011000 00001000 00000000	11111111 11111111 11111100 00000000

Să vedem acum ce se întâmplă când soseşte un pachet adresat pentru 194.24.17.4, care este reprezentat ca următorul şir de 32 de biţi :

11000010 00011000 00010001 00000100

Mai întâi, se face un ŞI logic cu masca de la Cambridge pentru a obţine:

11000010 00011000 00010000 00000000

Această valoare nu se potriveşte cu adresa de bază de la Cambridge, aşa că adresei originale i se aplică un ŞI logic cu masca de la Edinburgh pentru a se obţine:

11000010 00011000 00010000 00000000

Această valoare nu se potriveşte cu adresa de bază de la Edinburgh, aşa că se încearcă în continuare Oxford, obţinându-se:

11000010 00011000 00010000 00000000

Această valoare se potriveşte cu adresa de bază de la Oxford. Dacă în restul tabelei nu sunt găsite potriviri mai lungi, atunci este folosită intrarea pentru Oxford şi pachetul va fi trimis pe linia corespunzătoare.

Acum să privim cele trei universităţi din punctul de vedere al unui ruter din Omaha, Nebraska, care are doar patru linii de ieşire: Minneapolis, New York, Dallas şi Denver. Atunci când sof- tware-ul ruterului primeşte cele trei noi intrări, realizează că le poate combina pe toate trei într-o singură **intrare agregată** (eng,: **aggregate entry**) 194.24.0.0/19 cu adresa binară şi o masca de sub- reţea următoare :

11000010 00000000 00000000 00000000
11111111 11111111 11100000 00000000

Această intrare trimite toate pachetele destinate uneia dintre cele trei universităţi la New York. Reunind cele trei intrări ruterul din Omaha şi-a redus mărimea tabelei de rutare cu doi.

Dacă New York-ul are o singură conexiune cu Londra pentru tot traficul spre

Marea Britanie, poate folosi de asemenea o intrare agregată. Totuşi, dacă are două conexiuni separate pentru Londra şi Edinburgh, atunci trebuie să aibă trei intrări.

Ca o notă finală despre acest exemplu, intrarea agregată a ruterului din Omaha va trimite şi pachetele către adresele nealocate tot spre New York. Cât timp adresele sunt cu adevărat nealocate, aceasta nu contează pentru că aşa ceva nu ar trebui să se întâmple. Totuşi, dacă mai târziu sunt alocate unei companii din California, pentru a le trata va fi nevoie de o înregistrare separată, 194.24.12.0/22.

NAT – Translatarea adreselor de reţea

Adresele IP sunt insuficiente. Un ISP ar putea avea o adresă /16 (anterior de clasă B) oferindu-i 65.534 numere de staţii. Dacă are mai mulţi clienţi, are o problemă. Pentru utilizatorii casnici cu conexiuni pe linie telefonică (eng.: dial-up), o soluţie ar fi să se aloce dinamic o adresă IP fiecărui calculator în momentul în care sună şi se conectează şi să o dealoce în momentul în care se termină sesiunea. În acest fel o singură adresă /16 poate fi folosită pentru 65.534 utilizatori activi, ceea ce este probabil suficient pentru un ISP cu mai multe sute de mii de utilizatori. În momentul în care sesiunea se termină, adresa IP este alocată altui calculator care sună. Deşi această strategie funcţionează bine pentru un ISP cu un număr moderat de utilizatori casnici, eşuează pentru ISP-uri care deservesc preponderent companii.

Problema este că o companie se aşteaptă să fie on-line în mod continuu în timpul orelor de program. Atât companiile mici, cum ar fi o companie de turism compusă din trei persoane, cât şi mari corporaţii au mai multe calculatoare conectate de un LAN. Unele sunt calculatoarele personale ale angajaţilor; altele pot fi servere Web. În general în LAN există un ruter care este conectat cu ISP-ul printr-o linie închiriată pentru a avea conexiune continuă. Această situaţie impune ca fiecărui calculator să îi fie asociat un IP propriu pe parcursul întregii zile. De fapt, numărul total al calculatoarelor companiilor care sunt clienţi ai ISP-ului nu poate depăşi numărul de adrese IP pe care le posedă acesta. Pentru o adresă /16, aceasta limitează numărul total de calculatoare la 65.534. Pentru un ISP cu zeci de mii de companii cliente, această limită va fi repede depăşită.

Pentru a agrava situaţia, din ce în ce mai mulţi utilizatori se abonează de acasă la ADSL sau Internet prin cablu. Două dintre facilităţile oferite de aceste servicii sunt (1) utilizatorul primeşte o adresă IP permanentă şi (2) nu există taxă de conectare (doar o taxă lunară), aşa că mulţi utilizatori ai ADSL-ului şi ai Internetului prin cablu rămân conectaţi permanent. Aceasta contribuie la insuficienţa adreselor IP. Alocarea dinamică a adreselor IP aşa cum se face cu utilizatorii dial-up nu este utilă pentru că numărul adreselor IP folosite la un moment dat poate fi de multe ori mai mare decât numărul adreselor deţinute de ISP.

Şi pentru a complica şi mai mult lucrurile, mulţi utilizatori de ADSL şi Internet prin cablu au două sau mai multe calculatoare acasă, deseori câte unul pentru fiecare membru al familiei, şi toţi vor să fie online tot timpul folosind singura adresă IP care le-a fost alocată de către ISP. Soluţia este să se conecteze toate PC-urile printr-un LAN şi să se pună un ruter. Din punctul de vedere al ISP-ului, familia este acum ca o companie cu câteva calculatoare. Bine-aţi venit la Jones, Inc.

Problema epuizării adreselor IP nu este o problemă teoretică care ar putea apărea cândva în viitorul îndepărtat. A apărut aici şi acum. Soluţia pe termen lung este ca tot Internetul să migreze la IPv6, care are adrese de 128 de biţi. Tranziţia se

desfăşoară încet, dar vor trece ani până la finalizarea procesului. În consecinţă, anumite persoane au considerat că este nevoie de o rezolvare rapidă pe termen scurt. Rezolvarea a venit în forma NAT (Network Address Translation -

Translatarea adreselor de reţea), care este descrisă în RFC 3022 şi pe care o vom rezuma mai jos. Pentru informaţii suplimentare consultaţi (Dutcher, 2001).

Ideea de bază a NAT-ului este de a aloca fiecărei companii o singură adresă IP (sau cel mult un număr mic de adrese) pentru traficul Internet. În interiorul companiei, fiecare calculator primeşte o adresă IP unică, care este folosită pentru traficul intern. Totuşi, atunci când un pachet părăseşte compania şi se duce la ISP, are loc o translatare de adresă. Pentru a face posibil acest lucru, au fost declarate ca fiind private trei intervale de adrese IP. Companiile le pot folosi intern aşa cum doresc. Singura regulă este ca nici un pachet conţinând aceste adrese să nu apară pe Internet. Cele trei intervale rezervate sunt :

10.0.0.0	-	(16.777.216
172.16.0.	10.255.255.255/	gazde)
0	8	(1 .048.576
192.168		gazde)

Primul interval pune la dispoziţie 16.777.216 adrese (cu excepţia adreselor 0 şi -l, ca de obicei) şi este alegerea obişnuită a majorităţii companiilor, chiar dacă nu au nevoie de aşa de multe adrese.

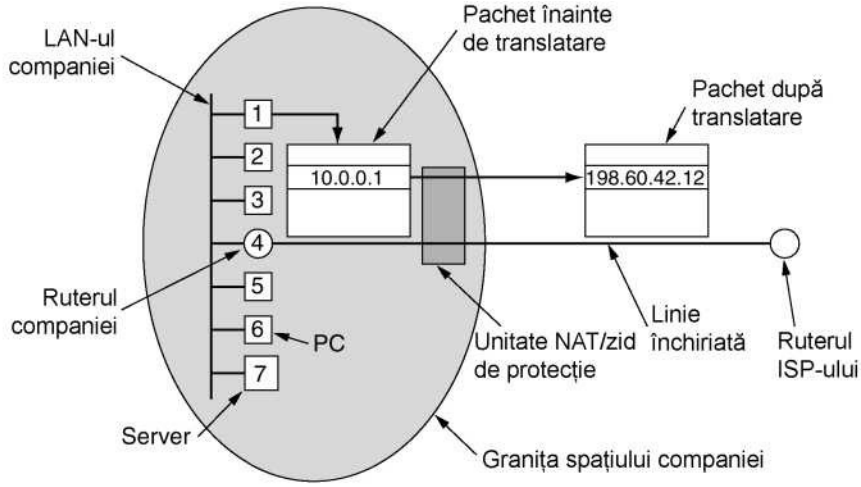


Fig. 5-60. Amplasarea şi funcţionarea unei

Funcţionarea NAT este ilustrată în fig. 5-60. În interiorul companiei fiecare maşină are o adresă unică de forma 10.x.y.z. Totuşi, când un pachet părăseşte compania, trece printr-o **unitate NAT** (eng.: **NAT box**)care converteşte adresa IP internă, 10.0.0.1 în figură, la adresa IP reală a companiei, 198.60.42.12, în acest exemplu. Unitatea NAT este deseori combinată într-un singur dispozitiv cu un zid de protecţie (eng.: firewall), care asigură securitatea controlând cu grijă ce intră şi iese din companie. Vom studia zidurile de protecţie în Cap. 8. De asemenea este posibilă integrarea unităţii NAT în ruterul companiei.

Până acum am trecut cu vederea un detaliu: atunci când vine răspunsul (de exemplu de la un server Web), este adresat 198.60.42.12, deci de unde ştie unitatea NAT cu care adresă să o înlocuiască pe aceasta? Aici este problema NAT-ului. Dacă ar fi existat un câmp liber în antetul IP, acel câmp ar fi putut fi folosit pentru a

memora cine a fost adevăratul emiţător, dar numai 1 bit este încă nefolosit. În principiu, o nouă opţiune ar putea fi creată pentru a reţine adevărata adresă a sursei, dar acest lucru ar necesita schimbarea codului din protocolul IP de pe toate staţiile din întregul Internet pentru a putea prelucra noua opţiune. Aceasta nu este o alternativă promiţătoare pentru o rezolvare rapidă.

În cele ce urmează se prezintă ceea ce s-a întâmplat cu adevărat. Proiectanţii NAT au observat că marea majoritate a pachetelor IP aveau conţinut TCP sau UDP. Atunci când vom studia protocoalele TCP şi UDP în Cap. 6, vom vedea că ambele au antete conţin un port sursă şi un port destinaţie. În continuare vom discuta doar despre porturi TCP, dar exact aceleaşi lucruri se aplică şi la UDP. Porturile sunt numere întregi pe 16 biţi care indică unde începe şi unde se termină o conexiune TCP. Aceste câmpuri pun la dispoziţie câmpul necesar funcţionării NAT.

Atunci când un proces doreşte să stabilească o conexiune TCP cu un proces de la distanţă, se ataşează unui port TCP nefolosit de pe maşina sa. Acesta se numeşte **portul sursă** şi spune codului TCP unde să trimită pachetele care vin şi aparţin acestei conexiuni. Procesul furnizează şi un **port destinaţie** pentru a spune cui să trimită pachetele în cealaltă parte. Porturile 0-1023 sunt rezervate pentru servicii bine cunoscute. De exemplu portul 80 este folosit de serverele Web, astfel încât clienţii să le poată localiza. Fiecare mesaj TCP care pleacă conţine atât un port sursă cât şi un port destinaţie. Cele două porturi permit identificarea proceselor care folosesc conexiunea la cele două capete.

S-ar putea ca o analogie să clarifice mai mult utilizarea porturilor. Imaginaţi-vă o companie cu un singur număr de telefon principal. Atunci când cineva sună la acest număr, vorbeşte cu un operator care întreabă ce interior doreşte şi apoi face legătura la acel interior. Numărul principal este analog cu adresa IP a companiei iar interioarele de la ambele capete sunt analoage cu porturile. Porturile reprezintă 16 biţi suplimentari de adresare identificând procesul care primeşte un pachet sosit.

Folosind câmpul *Port sursă* rezolvăm problema de corespondenţă. De fiecare dată când un pachet pleacă, el intră în unitatea NAT şi adresa sursă 10.x.y.z este înlocuită cu adresa reală a companiei. În plus, câmpul TCP *Port sursă* este înlocuit cu un index în tabela de translatare a unităţii NAT, care are 65.536 intrări. Această tabelă conţine adresa IP iniţială şi portul iniţial. În final, sunt recalculate şi inserate în pachet sumele de control ale antetelor IP şi TCP. Câmpul *Port sursă* trebuie înlocuit pentru că s-ar putea întâmpla, de exemplu, ca staţiile 10.0.0.1 şi 10.0.0.2 să aibă amândouă conexiuni care să folosească portul 5000, deci câmpul *Port sursă* nu este suficient pentru a identifica procesul emiţător.

Atunci când la unitatea NAT soseşte un pachet de la ISP, *Portul sursă* din antetul TCP este extras şi folosit ca index în tabela de corespondenţă a unităţii NAT. Din intrarea localizată sunt extrase şi inserate în pachet adresa IP internă şi *Portul sursă* TCP iniţial. Apoi sunt recalculate sumele de control TCP şi IP şi inserate în pachet. După aceea pachetul este transferat ruterului companiei pentru transmitere normală folosind adresa 10.x.y.z.

NAT poate fi folosit pentru rezolva insuficienţa adreselor IP pentru utilizatori de ADSL şi Internet prin cablu. Atunci când un ISP alocă fiecărui utilizator o adresă, foloseşte adrese 10.x.y.z. Atunci când pachete de la staţiile utilizatorilor ies din ISP şi intră în Internet trec printr-o cutie NAT care le translatează în adresele Internet

adevărate ale ISP-ului. La întoarcere pachetele trec prin maparea inversă. Din această perspectivă, pentru restul Internetului, ISP-ul şi utilizatorii săi ADSL/cablu arată la fel ca o mare companie.

Deşi această schemă rezolvă într-un fel problema, multe persoane din comunitatea IP o consideră un monstru pe faţa pământului. Rezumate succint, iată câteva dintre obiecţii. În primul rând, NAT violează modelul arhitectural al IP-ului, care afirmă că fiecare adresă IP identifică unic o singură maşină din lume. Întreaga structură software a Internetului este construită pornind de la acest fapt. Cu NAT, mii de maşini pot folosi (şi folosesc) adresa 10.0.0.1.

În al doilea rând, NAT schimbă natura Internetului de la o reţea fără conexiuni la un fel de reţea cu conexiuni. Problema este că o unitate NAT trebuie să menţină informaţii (corespondenţe) pentru fiecare conexiune care trece prin ea. Faptul că reţeaua păstrează starea conexiunilor este o proprietate a reţelelor orientate pe conexiune, nu a celor neorientate pe conexiune. Dacă o unitate NAT se defectează şi tabela de corespondenţă este pierdută, toate conexiunile TCP sunt distruse. În absenţa NAT, căderea ruterelor nu are nici un efect asupra TCP. Procesul care transmite ajunge la limita de timp în câteva secunde şi retransmite toate pachetele neconfirmate. Cu NAT, Internetul devine la fel de vulnerabil ca o reţea cu comutare de circuite.

În al treilea rând, NAT încalcă cea mai fundamentală regulă a protocoalelor pe niveluri: nivelul *k* nu poate face nici un fel de presupuneri referitor la ceea ce a pus nivelul *k+1* în câmpul de informaţie utilă. Principiul de bază este să se păstreze niveluri independente. Dacă mai târziu TCP este înlocuit de TCP-2, cu un antet diferit (de exemplu porturi pe 32 de biţi), NAT va eşua. Ideea protocoalelor pe niveluri este de a asigura că modificările la un nivel nu necesită modificări la celelalte niveluri. NAT distruge această independenţă.

În al patrulea rând, procesele din Internet nu sunt obligate să folosească TCP sau UDP. Dacă un utilizator de pe maşina *A* se decide să folosesc un nou protocol de transport pentru a comunica cu un utilizator de pe maşina *B* (de exemplu, pentru o aplicaţie multimedia), introducerea unei unităţi NAT va determina eşecul aplicaţiei, deoarece cutia NAT nu va fi în stare să localizeze corect câmpul TCP *Port sursă.*

În al cincilea rând, anumite aplicaţii introduc adrese IP în corpul mesajului. Receptorul extrage aceste adrese şi le foloseşte. De vreme ce NAT nu ştie nimic despre aceste adrese, nu le poate înlocui, deci orice încercare de a le folosi de către cealaltă parte va eşua. **FTP**, standardul **File Transfer Protocol** (rom.: protocol de transfer de fişiere) funcţionează în acest mod şi poate eşua în prezenţa NAT dacă nu se iau măsuri speciale. În mod similar protocolul pentru telefonie Internet H.323 (care va fi studiat în Cap. 8) are această proprietate şi nu va funcţiona în prezenţa NAT. Ar fi posibil să se modifice NAT-ul pentru a funcţiona cu H.323, dar modificarea codului unităţii NAT de fiecare dată când apare o aplicaţie nouă nu este o idee bună.

În al şaselea rând, deoarece câmpul TCP *Port sursă* are 16 biţi, o adresă IP poate fi pusă în corespondenţă cu cel mult 65.536 maşini. De fapt acest număr este puţin mai mic, deoarece primele 4096 porturi sunt rezervate pentru utilizări speciale. Totuşi dacă sunt disponibile mai multe adrese IP fiecare poate trata 61.440 maşini.

Acestea şi alte probleme ale NAT sunt discutate în RFC 2993. În general,

opozanţii NAT spun că rezolvând problema insuficienţei adreselor IP cu o soluţie temporară şi urâtă, presiunea pentru a implementa soluţia reală, adică tranziţia la IPv6, este redusă şi acesta este un lucru rău.

5.5.4 Protocoale de control în Internet

Pe lângă IP, care este folosit pentru transferul de date, Internet-ul are câteva protocoale de control la nivelul reţea, incluzând ICMP, ARP, RARP, BOOTP şi DHCP. În această secţiune vom arunca o privire asupra fiecăruia dintre ele.

Protocolul mesajelor de control din Internet

Operarea Internet-ului este strâns monitorizată de către rutere. Atunci când se întâmplă ceva neobişnuit, evenimentul este raportat prin **ICMP** (**Internet Control Message Protocol –** protocolul mesajelor de control din Internet), care este folosit şi pentru testarea Internet-ului. Sunt definite aproape o duzină de tipuri de mesaje ICMP. Cele mai importante sunt enumerate în fig. 5-61. Fiecare tip de mesaj ICMP este încapsulat într-un pachet IP.

Tipul mesajului	Descriere
Destinaţie inaccesibilă	Pachetul nu poate fi livrat
Timp depăşit	Câmpul timp de viaţă a ajuns la 0
Problemă de parametru	Câmp invalid în antet
Oprire sursă	Pachet de şoc
Redirectare	Învaţă un ruter despre geografie
Cerere de ecou	Întreabă o maşină dacă este activă
Răspuns ecou	Da, sunt activă
Cerere de amprentă de timp	La fel ca cererea de ecou, dar cu amprentă de
Răspuns cu amprentă de	La fel ca răspunsul ecou, dar cu amprentă de

Fig. 5-61. Tipurile principale de mesaje ICMP.

Mesajul DESTINAŢIE INACCESIBILĂ (DESTINATION UNREACHABLE) este folosit atunci când subreţeaua sau un ruter nu pot localiza destinaţia, sau un pachet cu bitul DF nu poate fi livrat deoarece o reţea cu „pachete mici” îi stă în cale.

Mesajul TIMP DEPĂŞIT (TIME EXCEEDED) este trimis când un pachet este eliminat datorită ajungerii contorului său la zero. Acest mesaj este un simptom al buclării pachetelor, al unei enorme congestii sau al stabilirii unor valori prea mici pentru ceas.

Mesajul PROBLEMĂ DE PARAMETRU (PARAMETER PROBLEM) indică detectarea unei valori nepermise într-un câmp din antet. Această problemă indică o eroare în programele IP ale gazdei emiţătoare sau eventual în programele unui ruter tranzitat.

Mesajul OPRIRE SURSĂ (SOURCE QUENCH) a fost folosit pe vremuri pentru a limita traficul gazdelor care trimiteau prea multe pachete. Când o gazdă primea acest mesaj, era de aşteptat să încetinească ritmul de transmisie. Este folosit arareori, deoarece când apare congestie, aceste pachete au tendinţa de a turna mai mult gaz pe foc. Controlul congestiei în Internet este făcut acum pe larg la nivelul transport şi va fi studiat în detaliu în Cap. 6.

Mesajul REDIRECTARE (REDIRECT) este folosit atunci când un ruter observă că un pachet pare a fi dirijat greşit. Este folosit de ruter pentru a spune gazdei

emiţătoare despre eroarea probabilă.

Mesajele CERERE ECOU (ECHO REQUEST) şi RĂSPUNS ECOU (ECHO REPLY) sunt folosite pentru a vedea dacă o anumită destinaţie este accesibilă şi activă. Este de aşteptat ca la recepţia mesajului ECOU, destinaţia să răspundă printr-un mesaj RĂSPUNS ECOU. Mesajele CERERE AMPRENTĂ DE TIMP (TIMESTAMP REQUEST) şi RĂSPUNS AMPRENTĂ DE TIMP (TIMESTAMP REPLY) sunt similare, cu excepţia faptului că în răspuns sunt înregistrate timpul de sosire a mesajului şi de plecare a răspunsului. Această facilitate este folosită pentru a măsura performanţele reţelei.

În plus faţă de aceste mesaje, au fost definite şi altele. Lista se află on-line la adresa www.iana. org/assignements/icmp-parameters.

Protocolul de rezoluţie a adresei: ARP

Deşi fiecare maşină din Internet are una sau mai multe adrese IP, acestea nu pot fi folosite de fapt pentru trimiterea pachetelor deoarece hardware-ul nivelului legăturii de date nu înţelege adresele Internet. Actualmente, cele mai multe gazde sunt ataşate la un LAN printr-o placă de interfaţă care înţelege numai adresele LAN. De exemplu, fiecare placă Ethernet fabricată până acum vine cu o adresă Ethernet de 48 biţi. Fabricanţii plăcilor Ethernet cer un spaţiu de adrese de la o autoritate centrală pentru a se asigura că nu există două plăci cu aceeaşi adresă (pentru a evita conflictele care ar apărea dacă cele două plăci ar fi în acelaşi LAN). Plăcile trimit şi primesc cadre pe baza adresei Ethernet de 48 biţi. Ele nu ştiu absolut nimic despre adresele IP pe 32 de biţi.

Se pune atunci întrebarea: Cum sunt transformate adresele IP în adrese la nivelul legăturii de date, ca de exemplu Ethernet? Pentru a explica care este funcţionarea, vom folosi exemplul din fig. 562, în care este ilustrată o universitate mică ce are câteva reţele de clasă C (denumită acum /24). Avem două reţele Ethernet, una în facultatea de Calculatoare, cu adresa IP 192.31.65.0 şi una în

facultatea de Inginerie Electrică, cu adresa IP 192.31.63.0. Acestea sunt conectate printr-un inel FDDI la nivelul campusului, care are adresa IP 192.31.60.0. Fiecare maşină dintr-o reţea Ethernet are o adresă Ethernet unică, etichetată de la E1 la E6, iar fiecare maşină de pe inelul FDDI are o adresă FDDI, etichetată de la F1 la F3.

Să începem prin a vedea cum trimite un utilizator de pe gazda 1 un pachet unui utilizator de pe gazda 2. Presupunem că expeditorul ştie numele destinatarului, ceva de genul *ary@eagle.cs.uni.edu*. Primul pas este aflarea adresei IP a gazdei 2, cunoscută ca *eagle.cs.uni.edu.* Această căutare este făcută de sistemul numelor de domenii (DNS), pe care îl vom studia în Cap. 7. Pentru moment, vom presupune că DNS-ul întoarce adresa IP a gazdei 2 (192.31.65.5).

Programele de la nivelurile superioare ale gazdei 1 construiesc un pachet cu 192.31.65.5 în câmpul *adresa destinatarului,* pachet care este trimis programelor IP pentru a-l transmite. Programele IP se uită la adresă şi văd că destinatarul se află în propria reţea, dar au nevoie de un mijloc prin care să determine adresa Ethernet a destinatarului. O soluţie este să avem undeva în sistem un fişier de configurare care transformă adresele IP în adrese Ethernet. Această soluţie este posibilă, desigur, dar pentru organizaţii cu mii de maşini, menţinerea fişierelor actualizate este o acţiune consumatoare de timp şi care poate genera erori.

O soluţie mai bună este ca gazda 1 să trimită un pachet de difuzare în reţeaua Ethernet întrebând: „Cine este proprietarul adresei IP 192.31.65.5?”. Pachetul de difuzare va ajunge la toate maşinile din reţeaua Ethernet 192.31.65.0 şi fiecare îşi va verifica adresa IP. Numai gazda 2 va răspunde cu adresa sa Ethernet *(E2).* În acest mod gazda 1 află că adresa IP 192.31.65.5 este pe gazda cu adresa Ethernet *E2.* Protocolul folosit pentru a pune astfel de întrebări şi a primi răspunsul se numeşte **ARP** (**Address Resolution Protocol –** Protocolul de rezoluţie a adresei). Aproape toate maşinile din Internet îl folosesc. El este definit în RFC 826.

Avantajul folosirii ARP faţă de fişierele de configurare îl reprezintă simplitatea. Administratorul de sistem nu trebuie să facă prea multe, decât să atribuie fiecărei maşini o adresă IP şi să hotărască măştile subreţelelor. ARP-ul face restul.

În acest punct, programele IP de pe gazda 1 construiesc un cadru Ethernet adresat către *E2*, pun pachetul IP (adresat către 193.31.65.5) în câmpul informaţie utilă şi îl lansează pe reţeaua Ethernet. Placa Ethernet a gazdei 2 detectează acest cadru, recunoaşte că este un cadru pentru ea, îl ia repede şi generează o întrerupere. Driverul Ethernet extrage pachetul IP din informaţia utilă şi îl trimite programelor IP, care văd că este corect adresat şi îl prelucrează.

Pentru a face ARP-ul mai eficient sunt posibile mai multe optimizări. Pentru început, la fiecare execuţie a ARP, maşina păstrează rezultatul pentru cazul în care are nevoie să contacteze din nou aceeaşi maşină în scurt timp. Data viitoare va găsi local corespondentul adresei, evitându-se astfel necesitatea unei a doua difuzări. În multe cazuri, gazda 2 trebuie să trimită înapoi un răspuns, ceea ce o forţează să execute ARP, pentru a determina adresa Ethernet a expeditorului. Această difuzare ARP poate fi evitată obligând gazda 1 să includă în pachetul ARP corespondenţa dintre adresa sa IP şi adresa Ethernet. Când pachetul ARP ajunge la gazda 2, perechea (192.31.65.7, E1) este memorată local de ARP pentru o folosire ulterioară.

De fapt, toate maşinile din reţeaua Ethernet pot memora această relaţie în memoria ARP locală.

Altă optimizare este ca fiecare maşină să difuzeze corespondenţa sa de adrese la pornirea maşinii. Această difuzare este realizată în general printr-un pachet ARP de căutare a propriei adrese IP. Nu ar trebui să existe un răspuns, dar un efect lateral al difuzării este introducerea unei înregistrări în memoria ascunsă ARP a tuturor. Dacă totuşi soseşte un răspuns (neaşteptat), înseamnă că două maşini au aceeaşi adresă IP. Noua maşină ar trebui să-l informeze pe administratorul de sistem şi să nu pornească.

Pentru a permite schimbarea relaţiei, de exemplu, când o placă Ethernet se strică şi este înlocuită cu una nouă (şi astfel apare o nouă adresă Ethernet), înregistrările din memoria ascunsă ARP ar trebui să expire după câteva minute.

Să privim din nou fig. 5-62, numai că de această dată gazda 1 vrea să trimită un pachet către gazda 4 (192.31.63.8). Folosirea ARP va eşua pentru că gazda 4 nu va vedea difuzarea (ruterele nu trimit mai departe difuzările de nivel Ethernet). Există două soluţii. Prima: ruterul facultăţii de Calculatoare poate fi configurat să răspundă la cererile ARP pentru reţeaua 193.31.63.0 (şi posibil şi pentru alte reţele locale). În acest caz, gazda 1 va memora local perechea (193.31.63.8, E3) şi va trimite tot traficul pentru gazda 4 către ruterul local. Această soluţie se numeşte **ARP cu intermediar** (**proxy ARP**). A doua soluţie este ca gazda 1 să-şi dea seama imediat că destinaţia se află pe o reţea aflată la distanţă şi să trimită tot traficul către o adresă Ethernet implicită care tratează tot traficul la distanţă, în acest caz *E3.* Această soluţie nu necesită ca ruterul facultăţii de Calculatoare să ştie ce reţele la distanţă deserveşte.

În ambele cazuri, ceea ce se întâmplă este că gazda 1 împachetează pachetul IP în câmpul informaţie utilă dintr-un cadru Ethernet adresat către *E3*. Când ruterul facultăţii de Calculatoare primeşte cadrul Ethernet, extrage pachetul IP din câmpul informaţie utilă şi caută adresa IP din tabelele sale de dirijare. Descoperă că pachetele pentru reţeaua 193.31.63.0 trebuie să meargă către ruterul 192.31.60.7. Dacă nu cunoaşte încă adresa FDDI a lui 193.31.60.7, difuzează un pachet ARP pe inel şi află că adresa din inel este *F3.* Apoi inserează pachetul în câmpul informaţie utilă al unui cadru FDDI adresat către *F3* şi îl transmite pe inel.

Driverul FDDI al ruterului facultăţii de Inginerie Electrică scoate pachetul din câmpul informaţie utilă şi îl trimite programelor IP care văd că trebuie să trimită pachetul către 192.31.63.8. Dacă această adresă IP nu este în memoria ascunsă ARP, difuzează o cerere ARP pe reţeaua Ethernet a facultăţii de Inginerie Electrică şi află că adresa destinaţie este *E6,* astfel încât construieşte un cadru Ethernet adresat către *E6*, pune pachetul în câmpul informaţie utilă şi îl trimite în reţeaua Ethernet. Când cadrul Ethernet ajunge la gazda 4, pachetul este extras din cadru şi trimis programelor IP pentru procesare.

Transferul între gazda 1 şi o reţea la distanţă peste un WAN funcţionează în esenţă asemănător, cu excepţia că de data aceasta tabelele ruterului facultăţii de Calculatoare îi vor indica folosirea ruterului WAN, a cărui adresă FDDI este *F2.*

RARP, BOOTP şi DHCP

ARP-ul rezolvă problema aflării adresei Ethernet corespunzătoare unei adrese IP date. Câteodată trebuie rezolvată problema inversă: dându-se o adresă Ethernet,

care este adresa IP corespunzătoare? În particular, această problemă apare când se porneşte o staţie de lucru fără disc. O astfel de maşină va primi, în mod normal, imaginea binară a sistemului său de operare de la un server de fişiere la distanţă. Dar cum îşi află adresa IP?

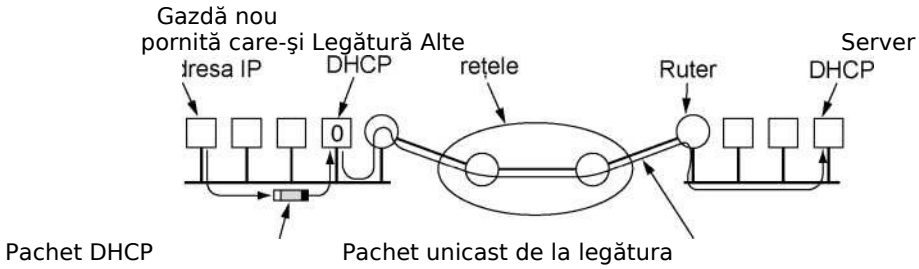
Prima soluţie proiectată a fost **RARP** (**Reverse Address Resolution Protocol –** Protocol de rezoluţie inversă a adresei) (definit în RFC 903). Acest protocol permite unei staţii de lucru de-abia pornită să difuzeze adresa sa Ethernet şi să spună: „Adresa mea Ethernet de 48 de biţi este 14.04.05.18.01.25. Ştie cineva adresa mea IP?" Serverul RARP vede această cerere, caută adresa Ethernet în fişierele sale de configurare şi trimite înapoi adresa IP corespunzătoare.

Folosirea RARP este mai bună decât introducerea unei adrese IP în imaginea de memorie, pentru că permite ca aceeaşi imagine să fie folosită pe toate maşinile. Dacă adresa IP ar fi fixată înăuntrul imaginii, atunci fiecare staţie de lucru ar necesita imaginea sa proprie.

Un dezavantaj al RARP este că, pentru a ajunge la serverul RARP, foloseşte o adresă destinaţie numai din l-uri (difuzare limitată). Cu toate acestea, asemenea difuzări nu sunt propagate de rutere, aşa încât este necesar un server RARP în fiecare reţea. Pentru a rezolva această problemă, a fost inventat un protocol alternativ de pornire, numit BOOTP (vezi RFC-urile 951, 1048 şi 1084). Spre deosebire de RARP, acesta foloseşte mesaje UDP, care sunt propagate prin rutere. De asemenea furnizează unei staţii de lucru fără disc informaţii suplimentare, care includ adresa IP a serverului de fişiere care deţine imaginea de memorie, adresa IP a ruterului implicit şi masca de subreţea care se foloseşte. BOOTP-ul este descris în RFC 951, 1048 şi 1084.

O problemă serioasă cu BOOTP este că necesită configurarea manuală a corespondenţelor între adresele IP şi adresele Ethernet. Atunci când o nouă gazdă este adăugată la LAN, nu poate folosi BOOTP decât atunci când un administrator îi alocă o adresă IP şi introduce manual (adresă Ethernet, adresă IP) în tabelele de configurare BOOTP. Pentru a elimina acest pas predispus la erori, BOOTP a fost extins şi i-a fost dat un nou nume: **DHCP** (**Dynamic Host Configuration Protocol –** Protocol dinamic de configurare a gazdei). DHCP permite atât atribuirea manuală de adrese IP, cât şi atribuirea automată. Este descris în RFC-urile 2131 şi 2132. În majoritatea sistemelor, a înlocuit în mare parte RARP şi BOOTP.

Ca şi RARP şi BOOTP, DHCP este bazat pe ideea unui server special care atribuie adrese IP gazdelor care cer una. Acest server nu trebuie să se afle în acelaşi LAN cu gazda care face cererea. Deoarece serverul DHCP s-ar putea să nu fie accesibil prin difuzare, este nevoie ca în fiecare LAN să existe un **agent de legătură DHCP** (**DHCP relay agent**), aşa cum se vede în fi fig. 5-63.



DISCOVER DHCP către serverul DHCP
(trimitere multiplă)

Fig. 5-63. Funcţionarea DHCP.

Pentru a-şi afla adresa IP, o maşină tocmai pornită difuzează un pachet DHCP DISCOVER. Agentul de legătură DHCP din LAN interceptează toate difuzările DHCP. Atunci când găseşte un pachet DHCP DISCOVER, îl trimite ca pachet unicast serverului DHCP, posibil într-o reţea depărtată. Singura informaţie de care are nevoie agentul este adresa IP a serverului DHCP.

O problemă care apare cu atribuirea automată a adreselor IP dintr-o rezervă comună este cât de mult ar trebui alocată o adresă IP . Dacă o gazdă părăseşte reţeaua şi nu returnează adresa sa IP serverului DHCP, acea adresă va fi pierdută permanent. După o perioadă de timp vor fi pierdute multe adrese. Pentru a preveni aceasta, atribuirea adresei IP va fi pentru o perioadă fixă de timp, o tehnică numită închiriere. Chiar înainte ca perioada de închiriere să expire, gazda trebuie să îi ceară DHCP-ului o reînnoire. Dacă nu reuşeşte să facă cererea sau dacă cererea este respinsă, gazda nu va mai putea folosi adresa IP care îi fusese dată mai devreme.

5.5.5 Protocolul de dirijare folosit de porţile interioare: OSPF

Am terminat acum studiul protocoalelor de control ale Internetului. Este timpul să trecem la următorul subiect : rutarea în Internet. Aşa cum am menţionat anterior, Internet-ul este construit dintr-un număr mare de sisteme autonome. Fiecare AS este administrat de o organizaţie diferită şi poate folosi propriul algoritm de dirijare în interior. De exemplu, reţelele interne ale companiilor *X, Y* şi *Z* ar fi văzute ca trei AS-uri dacă toate ar fi în Internet. Toate trei pot folosi intern algoritmi de dirijare diferiţi. Cu toate acestea, existenţa standardelor, chiar şi pentru dirijarea internă, simplifică implementarea la graniţele dintre AS-uri şi permite reutilizarea codului. În această secţiune vom studia dirijarea în cadrul unui AS. În următoarea, vom examina dirijarea între AS-uri. Un algoritm de dirijare în interiorul unui AS este numit **protocol de porţi interioare**; un algoritm de dirijare utilizat între AS-uri este numit **protocol de porţi exterioare**.

Protocolul de porţi interioare iniţial în Internet a fost un protocol de dirijare pe baza vectorilor distanţă (RIP) bazat pe algoritmul Bellman-Ford moştenit de la ARPANET. El funcţiona bine în sisteme mici, dar mai puţin bine pe măsură ce AS-urile s-au extins. El suferea de asemenea de problema numărării la infinit şi de o convergenţă slabă în general, aşa că în mai 1979 a fost înlocuit de un protocol bazat pe starea legăturilor. În 1988, Internet Engineering Task Force a început lucrul la un succesor. Acel succesor, numit **OSPF** (**Open Shortest Path First –** protocol public (deschis) bazat pe calea cea mai scurtă) a devenit un standard în 1990. Mulţi producători de rutere oferă suport pentru el şi acesta a devenit principalul protocol de porţi interioare. În cele ce urmează vom face o schiţă a funcţionării OSPF. Pentru informaţii complete, vedeţi RFC 2328.

Dată fiind experienţa îndelungată cu alte protocoale de dirijare, grupul care a proiectat noul protocol a avut o listă lungă de cerinţe care trebuiau satisfăcute. Mai întâi, algoritmul trebuia publicat în literatura publică (open), de unde provine „O”

din OSPF. O soluţie în proprietatea unei companii nu era un candidat. În al doilea rând, noul protocol trebuia să suporte o varietate de metrici de distanţă, incluzând distanţa fizică, întârzierea ş.a.m.d. În al treilea rând, el trebuia să fie un algoritm dinamic, care să se adapteze automat şi repede la schimbările în topologie.

În al patrulea rând şi nou pentru OSPF, trebuia să suporte dirijarea bazată pe tipul de serviciu. Noul protocol trebuia să fie capabil să dirijeze traficul de timp real într-un mod, iar alt tip de trafic în alt mod. Protocolul IP are câmpul *Tip serviciu,* dar nici un protocol de dirijare nu-l folosea. Acest câmp a fost introdus în OSPF dar tot nu îl folosea nimeni şi în cele din urmă a fost eliminat.

În al cincilea rând şi în legătură cu cele de mai sus, noul protocol trebuia să facă echilibrarea încărcării, divizând încărcarea pe mai multe linii. Majoritatea protocoalelor anterioare trimit toate pachetele pe cea mai bună cale. Calea de pe locul doi nu era folosită de loc. În multe cazuri, divizarea încărcării pe mai multe linii duce la performanţe mai bune.

În al şaselea rând, era necesar suportul pentru sisteme ierarhice. Până în 1988, Internet a crescut atât de mult, încât nu se poate aştepta ca un ruter să cunoască întreaga topologie. Noul protocol de dirijare trebuia să fie proiectat astfel, încât nici un ruter să nu fie nevoit să cunoască toată topologia.

În al şaptelea rând, se cerea un minim de măsuri de securitate, pentru a evita ca studenţii iubitori de distracţii să păcălească ruterele trimiţându-le informaţii de dirijare false. În fine, a fost necesară luarea de măsuri pentru a trata ruterele care au fost conectate la Internet printr-un tunel. Protocoalele precedente nu tratau bine acest caz.

OSPF suportă trei tipuri de conexiuni şi reţele:

1. Linii punct-la-punct între exact două rutere.
2. Reţele multiacces cu difuzare (de exemplu, cele mai multe LAN-uri).
3. Reţele multiacces fără difuzare (de exemplu, cele mai multe WAN-uri cu comutare de pachete).

O reţea **multiacces** este o reţea care poate să conţină mai multe rutere, fiecare dintre ele putând comunica direct cu toate celelalte. Toate LAN-urile şi WAN-urile au această proprietate. Fig. 5- 64(a) prezintă un AS care conţine toate cele trei tipuri de reţele. Observaţi că gazdele, în general, nu joacă un rol în OSPF.

OSPF funcţionează prin abstractizarea colecţiei de reţele, rutere şi linii reale într-un graf orientat în care fiecărui arc îi este atribuit un cost (distanţă, întârziere etc.). Apoi calculează cea mai scurtă cale bazându-se pe ponderile arcelor. O conexiune serială între două rutere este reprezentată de o pereche de arce, câte unul în fiecare direcţie. Ponderile lor pot fi diferite. O reţea multiacces este reprezentată de un nod pentru reţeaua însăşi plus câte un nod pentru fiecare ruter. Arcele de la nodul reţea la rutere au pondere 0 şi sunt omise din graf.

Fig. 5-64(b) prezintă graful pentru reţeaua din fig. 5-64(a). Ponderile sunt simetrice, dacă nu se specifică altfel. Fundamental, ceea ce face OSPF este să reprezinte reţeaua reală printr-un graf ca acesta şi apoi să calculeze cea mai scurtă cale de la fiecare ruter la fiecare alt ruter.

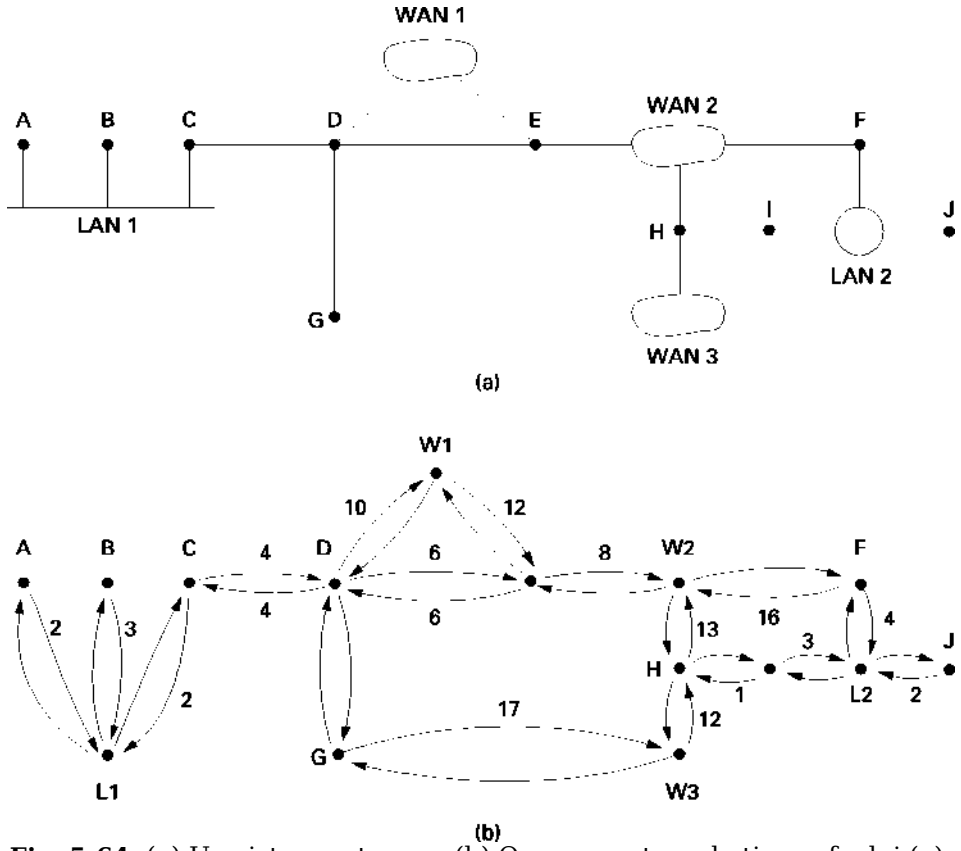


Fig. 5-64. (a) Un sistem autonom. (b) O reprezentare de tip graf a lui (a).

Multe din AS-urile din Internet sunt foarte mari şi nu sunt simplu de administrat. OSPF le permite să fie divizate în **zone** numerotate, unde o zonă este o reţea sau o mulţime de reţele învecinate. Zonele nu se suprapun şi nu este necesar să fie exhaustive, în sensul că unele rutere pot să nu aparţină nici unei zone. O zonă este o generalizare a unei subreţele. În afara zonei, topologia şi detaliile sale nu sunt vizibile.

Orice AS are o zonă **de coloană vertebrală**, numită zona 0. Toate zonele sunt conectate la coloana vertebrală, eventual prin tunele, astfel încât este posibil să se ajungă din orice zonă din AS în orice altă zonă din AS prin intermediul coloanei vertebrale. Un tunel este reprezentat în graf ca un arc şi are un cost. Fiecare ruter care este conectat la două sau mai multe zone aparţine coloanei vertebrale. Analog cu celelalte zone, topologia coloanei vertebrale nu este vizibilă din afara coloanei vertebrale.

În interiorul unei zone, fiecare ruter are aceeaşi bază de date pentru starea legăturilor şi foloseşte acelaşi algoritm de cea mai scurtă cale. Principala sa sarcină este să calculeze cea mai scurtă cale de la sine la fiecare alt ruter din zonă, incluzând ruterul care este conectat la coloana vertebrală, din care trebuie să existe cel puţin unul. Un ruter care conectează două zone are nevoie de bazele de date pentru ambele zone şi trebuie să folosească algoritmul de cale cât mai scurtă separat pentru fiecare zonă.

În timpul operării normale pot fi necesare trei tipuri de căi: intrazonale, interzonale şi interAS- uri. Rutele intrazonale sunt cele mai uşoare, din moment ce ruterul sursă ştie întotdeauna calea cea

mai scurtă spre ruterul destinaţie. Dirijarea interzonală se desfăşoară întotdeauna în trei paşi: drum de la sursă la coloana vertebrală; drum de-a lungul coloanei vertebrale până la zona destinaţie; drum la destinaţie. Acest algoritm forţează o configuraţie de tip stea pentru OSPF, coloana vertebrală fiind concentratorul (hub), iar celelalte zone fiind spiţele. Pachetele sunt dirijate de la sursă la destinaţie „ca atare". Ele nu sunt încapsulate sau trecute prin tunel, cu excepţia cazului în care merg spre o zonă a cărei unică conexiune la coloana vertebrală este un tunel. Fig. 5-65 arată o parte a Internet- ului cu AS-uri şi zone.

OSPF distinge patru clase de rutere:

1. Ruterele interne sunt integral în interiorul unei zone.
2. Ruterele de la graniţa zonei conectează două sau mai multe zone.
3. Ruterele coloanei vertebrale sunt pe coloana vertebrală.
4. Ruterele de la graniţa AS-urilor discută cu ruterele din alte AS-uri.

Aceste clase pot să se suprapună. De exemplu, toate ruterele de graniţă fac parte în mod automat din coloana vertebrală. În plus, un ruter care este în coloana vertebrală, dar nu face parte din nici o altă zonă este de asemenea un ruter intern. Exemple din toate cele patru clase de rutere sunt ilustrate în fig. 5-65.

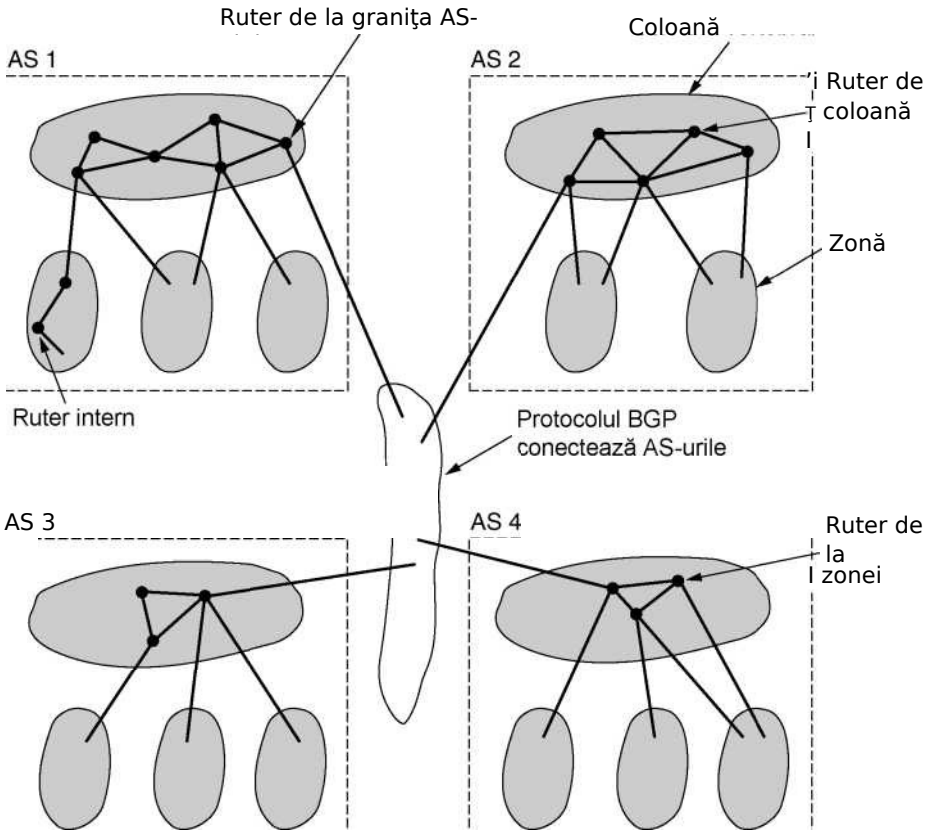


Fig. 5-65. Relaţia dintre AS-uri, coloane vertebrale şi

Când un ruter porneşte, trimite mesaje HELLO pe toate liniile sale punct-la-punct şi trimite multiplu (multicast) în LAN-urile grupului compus din toate celelalte rutere. În WAN-uri, are nevoie de anumite informaţii de configuraţie, pentru a şti pe cine să contacteze. Din răspunsuri, fiecare ruter află care sunt vecinii săi. Ruterele din acelaşi LAN sunt toate vecine.

OSPF funcţionează prin schimb de informaţii între rutere **adiacente**, care nu este acelaşi lucru cu schimbul de informaţii între ruterele vecine. În particular, este ineficient ca fiecare ruter dintr-un LAN să discute cu fiecare alt ruter din LAN. Pentru a evita această situaţie, un ruter este ales ca **ruter desemnat**. Se spune că el este adiacent cu toate celelalte rutere din LAN-ul său şi schimbă informaţii cu ele. Ruterele vecine care nu sunt adiacente nu schimbă informaţii între ele. De asemenea, este actualizat în permanenţă şi un ruter desemnat de rezervă pentru a uşura tranziţia dacă ruterul desemnat primar se defectează şi trebuie să fie înlocuit imediat.

În timpul funcţionării normale, fiecare ruter inundă periodic cu mesaje ACTUALIZARE STARE LEGĂTURĂ (Link State Update) fiecare ruter adiacent. Acest mesaj indică starea sa şi furnizează costurile folosite în baza de date topologică. Mesajele de inundare sunt confirmate pentru a le face sigure. Fiecare mesaj are un număr de secvenţă, astfel încât un ruter poate vedea dacă un mesaj ACTUALIZARE STARE LEGĂTURĂ este mai vechi sau mai nou decât ceea ce are deja. De asemenea, ruterele trimit aceste mesaje când o linie cade sau îşi revine sau când costul acesteia se modifică.

Mesajele DESCRIERE BAZA DE DATE (Database Description) dau numerele de secvenţă pentru toate intrările de stare a liniei deţinute actual de emiţător. Prin compararea valorilor proprii cu acelea ale emiţătorului, receptorul poate determina cine are cea mai recentă valoare. Aceste mesaje sunt folosite când o linie este refăcută.

Fiecare partener poate cere informaţii de stare a legăturii de la celălalt folosind mesaje CERERE STARE LEGĂTURĂ (Link State Request). Rezultatul concret al acestui algoritm este că fiecare pereche de rutere adiacente verifică să vadă cine are cele mai recente date şi astfel, noua informaţie este răspândită în zonă. Toate aceste mesaje sunt trimise ca simple pachete IP. Cele cinci tipuri de mesaje sunt rezumate în fig. 5-66.

Tip mesaj	Descriere
Hello	Folosit pentru descoperirea vecinilor
Actualizare Stare Legătură	Emiţătorul furnizează vecinilor săi costurile
Confirmare Stare Legătură	Confirmă actualizarea stării legăturii
Descriere Bază de Date	Anunţă ce actualizări are emiţătorul
Cerere Stare Legătură	Cere informaţii de la partener

Fig. 5-66. Cele cinci tipuri de mesaje OSPF.

În final, putem să asamblăm toate piesele. Folosind inundarea, fiecare ruter informează toate celelalte rutere din zona sa despre vecinii şi costurile sale. Această informaţie permite fiecărui ruter să construiască graful zonei (zonelor) sale şi să calculeze cea mai scurtă cale. Zona de coloană vertebrală face şi ea acelaşi lucru. În plus, ruterele de coloană vertebrală acceptă informaţia de la ruterele zonei de graniţă cu scopul de a calcula cea mai bună cale de la fiecare ruter de coloană

vertebrală către fiecare alt ruter. Această informaţie este propagată înapoi către ruterele zonei de graniţă, care o fac publică în zonele lor. Folosind această informaţie, un ruter gata să trimită un pachet interzonal poate selecta cel mai bun ruter de ieşire către coloana vertebrală.

5.6.5 Protocolul de dirijare pentru porţi externe: BGP

În cadrul unui singur AS, protocolul de dirijare recomandat este OSPF (deşi, desigur, nu este singurul folosit). Între AS-uri se foloseşte un protocol diferit, **BGP** (**Border Gateway Protocol –** Protocolul porţilor de graniţă). Între AS-uri este necesar un protocol diferit pentru că scopurile unui protocol pentru porţi interioare şi ale unui protocol pentru porţi exterioare sunt diferite. Tot ce trebuie să facă un protocol pentru porţi interioare este să mute pachetele cât mai eficient de la sursă la destinaţie. El nu trebuie să-şi facă probleme cu politica.

Ruterele ce folosesc protocolul de porţi exterioare trebuie să ţină cont într-o mare măsură de politică (Metz, 2001). De exemplu, un AS al unei corporaţii poate dori facilitatea de a trimite pachete oricărui sit Internet şi să recepţioneze pachete de la orice sit Internet. Cu toate acestea, poate să nu dorească să asigure tranzitarea pentru pachetele originare dintr-un AS străin destinate unui AS străin diferit, chiar dacă prin AS-ul propriu trece cea mai scurtă cale dintre cele două AS-uri străine („Asta este problema lor, nu a noastră.”). Pe de altă parte, poate fi dispus să asigure tranzitarea pentru vecinii săi, sau chiar pentru anumite AS-uri care au plătit pentru acest serviciu. Companiile telefonice, de exemplu, pot fi fericite să acţioneze ca un purtător pentru clienţii lor, dar nu şi pentru alţii. Protocoalele pentru porţi externe, în general şi BGP în particular, au fost proiectate pentru a permite forţarea multor tipuri de politici de dirijare pentru traficul între AS-uri.

Politicile tipice implică consideraţii politice, de securitate sau economice. Câteva exemple de constrângeri de dirijare sunt:

1. Nu se tranzitează traficul prin anumite AS-uri.
2. Nu se plasează Irak-ul pe o rută care porneşte din Pentagon.
3. Nu se folosesc Statele Unite pentru a ajunge din Columbia Britanică în Ontario.
4. Albania este tranzitată numai dacă nu există altă alternativă către destinaţie.
5. Traficul care pleacă sau ajunge la IBM nu trebuie să tranziteze Microsoft.

În mod obişnuit politicile sunt configurate manual în fiecare ruter BGP (sau sunt incluse folosind un anumit tip de script). Ele nu sunt parte a protocolului însuşi.

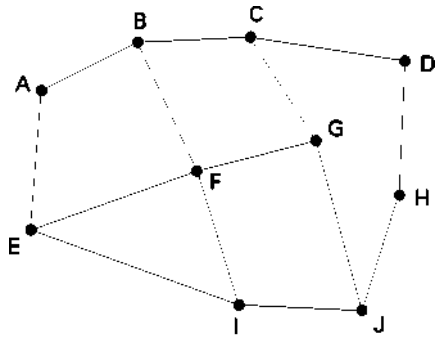
Din punctul de vedere al unui ruter BGP, lumea constă din AS-uri şi liniile care le conectează. Două AS-uri sunt considerate conectate dacă există o linie între două rutere de graniţă din fiecare. Dat fiind interesul special al BGP-ului pentru traficul în tranzit, reţelele sunt grupate în trei categorii. Prima categorie este cea a **reţelelor ciot (stub networks)**, care au doar o conexiune la graful BGP. Acestea nu pot fi folosite pentru traficul în tranzit pentru că nu este nimeni la capătul celălalt. Apoi vin **reţelele multiconectate**. Acestea pot fi folosite pentru traficul în tranzit, cu excepţia a ceea ce ele refuză. În final, sunt **reţele de tranzit**, cum ar fi coloanele vertebrale, care sunt doritoare să manevreze pachetele altora, eventual cu unele restricţii, şi de obicei pentru o plată.

Perechile de rutere BGP comunică între ele stabilind conexiuni TCP. Operarea în acest mod oferă comunicaţie sigură şi ascunde toate detaliile reţelelor traversate.

BGP este la bază un protocol bazat pe vectori distanţă, dar destul de diferit de majoritatea celorlalte cum ar fi RIP. În loc să menţină doar costul până la fiecare destinaţie, fiecare ruter BGP memorează calea exactă folosită. Similar, în loc să trimită periodic fiecărui vecin costul său estimat către fiecare destinaţie posibilă, fiecare ruter BGP comunică vecinilor calea exactă pe care o foloseşte.

Ca exemplu, să considerăm ruterele BGP din fig. 5-67(a). În particular, să considerăm tabela de dirijare a lui *F.* Să presupunem că el foloseşte calea *FGCD* pentru a ajunge la *D.* Când vecinii îi dau

informaţiile de dirijare, ei oferă căile lor complete, ca în fig. 5-67(b) (pentru simplitate, este arătată doar destinaţia *D).*



Informaţia despre D pe care F o primeşte de la vecini

De la B: "Eu folosesc BCD" De la G: "Eu folosesc GCD" De la I: "Eu folosesc IFGCD"
De la E: "Eu folosesc EFGCD"

(b

Fig. 5-67. (a) O mulţime de rutere BGP. (b) Informaţia trimisă lui *F.*

După ce sosesc toate căile de la vecini, *F* le examinează pentru a vedea care este cea mai bună. El elimină imediat căile de la *I* şi *E,* din moment ce aceste căi trec chiar prin F. Alegerea este apoi între B şi G. Fiecare ruter BGP conţine un modul care examinează căile către o destinaţie dată şi le atribuie scoruri, întorcând, pentru fiecare cale, o valoare pentru „distanţa" către acea destinaţie. Orice cale care violează o constrângere politică primeşte automat un scor infinit. Apoi, ruterul adoptă calea cu cea mai scurtă distanţă. Funcţia de acordare a scorurilor nu este parte a protocolului BGP şi poate fi orice funcţie doresc administratorii de sistem.

BGP rezolvă uşor problema numără-la-infinit care chinuie alţi algoritmi bazaţi pe vectori distanţă. De exemplu, să presupunem că *G* se defectează sau că linia *FG* cade. Atunci *F* primeşte căi de la ceilalţi trei vecini rămaşi. Aceste rute sunt *BCD, IFGCD* şi *EFGCD.* El poate observa imediat că ultimele două căi sunt inutile, din moment ce trec chiar prin *F*, aşa că alege *FBCD* ca noua sa cale. Alţi algoritmi bazaţi pe vectori distanţă fac de multe ori alegerea greşită, pentru că ei nu pot spune care din vecinii lor au căi independente către destinaţie şi care nu. Definirea BGP-ului se găseşte în RFC 1771 şi 1774.

5.6.6 Trimiterea multiplă în Internet

Comunicaţia IP normală este între un emiţător şi un receptor. Cu toate acestea, pentru unele aplicaţii, este util ca un proces să fie capabil să trimită simultan unui număr mare de receptori. Exemple sunt actualizarea bazelor de date distribuite multiplicate, transmiterea cotărilor de la bursă mai multor agenţi de bursă, tratarea convorbirilor telefonice de tipul conferinţelor digitale (aşadar cu mai mulţi participanţi).

IP-ul suportă trimiterea multiplă (multicast), folosind adrese de clasă D. Fiecare adresă de clasă D identifică un grup de gazde. Pentru identificarea grupurilor sunt

disponibili douăzeci şi opt de biţi, aşa încât pot exista în acelaşi moment peste 250 milioane de grupuri. Când un proces trimite un pachet unei adrese de clasă D, se face cea mai bună încercare pentru a-l livra tuturor membrilor grupului adresat, dar nu sunt date garanţii. Unii membri pot să nu primească pachetul.

Sunt suportate două tipuri de adrese de grup: adrese permanente şi adrese temporare. Un grup permanent există întotdeauna şi nu trebuie configurat. Fiecare grup permanent are o adresă de grup permanentă. Câteva exemple de adrese de grup permanente sunt:

224.0.0.1 Toate sistemele dintr-un LAN.
224.0.0.2 Toate ruterele dintr-un LAN.
224.0.0.5 Toate ruterele OSPF dintr-un LAN.
224.0.0.6 Toate ruterele desemnate dintr-un LAN.

Grupurile temporare trebuie create înainte de a fi utilizate. Un proces poate cere gazdei sale să intre într-un anume grup. De asemenea, el poate cere gazdei sale să părăsească grupul. Când ultimul proces părăseşte un grup, acel grup nu mai este prezent pe calculatorul său gazdă. Fiecare gazdă memorează căror grupuri aparţin la un moment dat procesele sale.

Trimiterea multiplă este implementată de rutere speciale de trimitere multiplă, care pot sau nu coabita cu ruterele standard. Cam o dată pe minut, fiecare ruter de trimitere multiplă face o trimitere multiplă hardware (la nivel legătură de date) pentru gazdele din LAN-ul său (adresa 224.0.0.1), cerându-le să raporteze căror grupuri aparţin procesele lor la momentul respectiv. Fiecare gazdă trimite înapoi răspunsuri pentru toate adresele de clasă D de care este interesată.

Aceste pachete de întrebare şi răspuns folosesc un protocol numit **IGMP** (**Internet Group Management Protocol –** Protocol de gestiune a grupurilor Internet), care se aseamănă întrucâtva cu ICMP. El are numai două tipuri de pachete: întrebare şi răspuns, fiecare cu un format fix, simplu, care conţine unele informaţii de control în primul cuvânt al câmpului informaţie utilă şi o adresă de clasă D în al doilea cuvânt. El este descris în RFC 1112.

Dirijarea cu trimitere multiplă este realizată folosind arbori de acoperire. Fiecare ruter de dirijare multiplă schimbă informaţii cu vecinii săi, folosind un protocol modificat bazat pe vectori distanţă cu scopul ca fiecare să construiască pentru fiecare grup un arbore de acoperire care să acopere toţi membrii grupului. Pentru a elimina ruterele şi reţelele neinteresate de anumite grupuri, se utilizează diverse optimizări de reducere a arborelui. Pentru evitarea deranjării nodurilor care nu fac parte din arborele de acoperire, protocolul foloseşte intensiv trecerea prin tunel.

5.6.7 IP mobil

Mulţi utilizatori ai Internet-ului au calculatoare portabile şi vor să rămână conectaţi la Internet atunci când vizitează un sit Internet aflat la distanţă, şi chiar şi pe drumul dintre cele două. Din nefericire, sistemul de adresare IP face lucrul la depărtare de casă mai uşor de zis decât de făcut. În această secţiune vom examina problema şi soluţia. O descriere mai detaliată este dată în (Perkins, 1998a).

Problema apare chiar în schema de adresare. Fiecare adresă IP conţine un număr de reţea şi un număr de gazdă. De exemplu, să considerăm maşina cu adresa IP 160.80.40.20/16. Partea 160.80 indică numărul de reţea (8272 în notaţie zecimală). Ruterele din toată lumea au tabele de dirijare care spun ce linie se foloseşte pentru a ajunge la reţeaua 160.80. Oricând vine un pachet cu adresa IP destinaţie de forma 160.80.xxx.yyy, pachetul pleacă pe respectiva linie.

Dacă, dintr-o dată, maşina cu adresa respectivă este transferată într-un alt loc din Internet, pachetele vor continua să fie dirijate către LAN-ul (sau ruterul) de acasă. Proprietarul nu va mai primi poştă electronică şi aşa mai departe. Acordarea unei noi adrese IP maşinii, adresă care să corespundă cu noua sa locaţie, nu este atractivă pentru că ar trebui să fie informate despre schimbare un mare număr de persoane, programe şi baze de date.

O altă abordare este ca ruterele să facă dirijarea folosind adresa IP completă, în loc să folosească numai adresa reţelei. Cu toate acestea, această strategie ar necesita ca fiecare ruter să aibă milioane de intrări în tabele, la un cost astronomic pentru Internet.

Când oamenii au început să ceară posibilitatea de a-şi conecta calculatoarele portabile oriunde sar duce, IETF a constituit un Grup de Lucru pentru a găsi o soluţie. Grupul de Lucru a formulat rapid un număr de obiective considerate necesare în orice soluţie. Cele majore au fost:

1. Fiecare gazdă mobilă trebuie să fie capabilă sa folosească adresa sa IP de baza oriunde.
2. Nu au fost permise schimbări de programe pentru gazdele fixe.
3. Nu au fost permise schimbări pentru programele sau tabelele ruterelor.
4. Cele mai multe pachete pentru gazdele mobile nu ar trebui să facă ocoluri pe drum.
5. Nu trebuie să apară nici o suprasolicitare când o gazdă mobilă este acasă.

Soluţia aleasă este cea descrisă în secţiunea 5.2.8. Pentru a o recapitula pe scurt, fiecare sit care doreşte să permită utilizatorilor săi să se deplaseze trebuie să asigure un agent local. Fiecare sit care doreşte să permită accesul vizitatorilor trebuie să creeze un agent pentru străini. Când o gazdă mobilă apare într-un sit străin, ea contactează gazda străină de acolo şi se înregistrează. Gazda străină contactează apoi agentul local al utilizatorului şi îi dă o **adresă a intermediarului**, în mod normal adresa IP proprie a agentului pentru străini.

Când un pachet ajunge în LAN-ul de domiciliu al utilizatorului, el vine la un ruter ataşat la LAN. Apoi ruterul încearcă să localizeze gazda în mod uzual, prin difuzarea unui pachet ARP întrebând, de exemplu: „Care este adresa Ethernet a lui 160.80.40.20?” Agentul local răspunde la această întrebare dând propria adresă Ethernet. Apoi ruterul trimite pachetele pentru 160.80.40.20 la agentul local. El, în schimb, le trimite prin tunel la adresa intermediarului prin încapsularea lor în câmpul informaţie utilă al unui pachet IP adresat agentului pentru străini. După aceasta, agentul pentru străini le desface şi le livrează la adresa de nivel legătură de date a gazdei mobile. În plus, agentul local dă emiţătorului adresa intermediarului, aşa încât viitoarele pachete pot fi trimise prin tunel direct la agentul pentru străini. Această soluţie răspunde tuturor cerinţelor expuse mai sus.

Probabil că merită menţionat un mic amănunt. În momentul în care gazda mobilă se mută, probabil că ruterul are adresa ei Ethernet (care în curând va fi invalidă) memorată în memoria ascunsă. Pentru a înlocui această adresă Ethernet cu cea a agentului local, se foloseşte un truc numit **ARP gratuit**. Acesta este un mesaj special, nesolicitat, către ruter pe care îl determină să schimbe o anumită intrare din memoria ascunsă, în acest caz cea a gazdei mobile care urmează să plece. Când, mai târziu, gazda mobilă se întoarce, este folosit acelaşi truc pentru a actualiza din nou memoria ascunsă a ruterului.

Nu există nimic în proiect care să împiedice o gazdă mobilă să fie propriul său agent pentru străini, dar această abordare funcţionează numai dacă gazda mobilă (în postura sa de agent pentru străini) este conectată logic în Internet la situl său curent. De asemenea, ea trebuie să fie capabilă să obţină pentru folosire o adresă (temporară) de intermediar. Acea adresă IP trebuie să aparţină LAN-ului în care este ataşată în mod curent.

Soluţia IETF pentru gazde mobile rezolvă un număr de alte probleme care nu au fost menţionate până acum. De exemplu, cum sunt localizaţi agenţii? Soluţia este ca fiecare agent să-şi difuzeze periodic adresa şi tipul de serviciu pe care doreşte să-l ofere (de exemplu, agent local, pentru străini sau amândouă). Când o gazdă mobilă ajunge undeva, ea poate asculta aşteptând aceste difuzări, numite **anunţuri**. Ca o alternativă, ea poate difuza un pachet prin care îşi anunţă sosirea şi să spere că agentul pentru străini local îi va răspunde.

O altă problemă care a trebuit rezolvată este cum să se trateze gazdele mobile nepoliticoase, care pleacă fără să spună la revedere. Soluţia este ca înregistrarea să fie valabilă doar pentru un interval de timp fixat. Dacă nu este reîmprospătată periodic, ea expiră şi ca urmare gazda străină poate să-şi cureţe tabelele.

O altă problemă este securitatea. Când un agent local primeşte un mesaj care-i cere să fie amabil să retrimită toate pachetele Robertei la o anume adresă IP, ar fi mai bine să nu se supună decât dacă este convins că Roberta este sursa acestei cereri şi nu altcineva încercând să se dea drept ea. În acest scop sunt folosite protocoale criptografice de autentificare. Vom studia asemenea protocoale în cap. 8.

Un punct final adresat de Grupul de Lucru se referă la nivelurile de mobilitate. Să ne imaginăm un avion cu o reţea Ethernet la bord folosită de către calculatoarele de navigare şi de bord. În această reţea Ethernet există un ruter standard, care discută cu Internet-ul cablat de la sol printr-o legătură radio. Într-o bună zi, unui director de marketing isteţ îi vine ideea să instaleze conectoare Ethernet în toate braţele fotoliilor, astfel încât şi pasagerii cu gazde mobile să se poată conecta.

Acum avem două niveluri de mobilitate: calculatoarele proprii ale aeronavei, care sunt staţionare în raport cu reţeaua Ethernet şi calculatoarele pasagerilor, care sunt mobile în raport cu ea. În plus, ruterul de la bord este mobil în raport cu ruterele de la sol. Mobilitatea în raport cu un sistem care este la rândul său mobil este tratată folosind recursiv tunele.

5.6.8 IPv6

În timp ce CIDR şi NAT îi mai pot acorda câţiva ani, toată lumea îşi dă seama că

zilele IP-ului în forma curentă (IPv4) sunt numărate. În plus faţă de aceste probleme tehnice, există un alt aspect întrezărit în fundal. La începuturile sale, Internet-ul a fost folosit în mare măsură de universităţi, industria de vârf şi de guvernul Statelor Unite (în mod special de Departamentul Apărării). O dată cu explozia interesului faţă de Internet începând de la mijlocul anilor 1990, a început să fie utilizat de un grup diferit de persoane, în special persoane cu cerinţe diferite. Pe de o parte, numeroase persoane cu calculatoare portabile fără fir îl folosesc pentru a ţine legătura cu baza de acasă. Pe de altă parte, o dată cu iminenta convergenţă a industriilor calculatoarelor, comunicaţiilor şi a distracţiilor, s-ar putea să nu mai fie mult până când fiecare telefon sau televizor din lume va fi un nod Internet, producând un miliard de maşini folosite pentru audio şi video la cerere. În aceste condiţii, a devenit clar că IP-ul trebuie să evolueze şi să devină mai flexibil.

Observând aceste probleme la orizont, IETF a început să lucreze în 1990 la o nouă versiune de IP, una care să nu îşi epuizeze niciodată adresele, să rezolve o gamă largă de alte probleme şi să fie totodată mai flexibilă şi mai eficientă. Obiectivele majore au fost:

1. Să suporte miliarde de gazde, chiar cu alocare ineficientă a spaţiului de adrese.
2. Să reducă dimensiunea tabelelor de dirijare.
3. Să simplifice protocolul, pentru a permite ruterelor să proceseze pachetele mai rapid.
4. Să asigure o securitate mai bună (autentificare şi confidenţialitate) faţă de IP-ul curent.
5. Să acorde o mai mari atenţie tipului de serviciu, în special pentru datele de timp real.
6. Să ajute trimiterea multiplă, permiţând specificarea de domenii.
7. Să creeze condiţiile pentru ca o gazdă să poată migra fără schimbarea adresei sale.
8. Să permită evoluţia protocolului în viitor.
9. Să permită coexistenţa noului şi vechiului protocol pentru câţiva ani.

Pentru a găsi un protocol care să îndeplinească toate aceste cerinţe, IETF a emis o cerere de propuneri şi discuţii în RFC 1550. Au fost primite douăzeci şi unu de răspunsuri, nu toate din ele propuneri complete. Până în decembrie 1992, au ajuns pe masa discuţiilor şapte propuneri serioase. Ele variau de la efectuarea de mici cârpeli la IP până la renunţarea completă la el şi înlocuirea cu un protocol complet nou.

O propunere a fost folosirea TCP peste CLNP, care cu cei 160 de biţi de adresă ai săi ar fi oferit spaţiu de adrese suficient pentru totdeauna şi ar fi unificat două protocoale majore de nivel reţea. Cu toate acestea, multe persoane au simţit că aceasta ar fi fost o acceptare a faptului că, de fapt, a fost făcut ceva chiar bine în lumea OSI, o afirmaţie considerată incorectă din punct de vedere politic în cercurile Internet. CLNP a fost modelat apropiat de IP, aşa încât cele două nu sunt chiar atât de diferite. De fapt, protocolul care a fost ales în final diferă de IP cu mult mai mult decât diferă CLNP. O altă lovitură împotriva CLNP a fost slabul suport pentru tipuri de servicii, ceva necesar pentru transmiterea eficientă de multimedia.

Trei din cele mai bune propuneri au fost publicate în *IEEE Network* (Deering, 1993; Francis, 1993 şi Katz şi Ford, 1993). După multe discuţii, revizii şi manevre de culise, a fost selectată o versiune combinată modificată a propunerilor lui Deering şi Francis, până atunci numită **SIPP** (**Simple Internet Protocol Plus** – Protocol simplu, îmbunătăţit, pentru Internet) şi i s-a dat numele de **IPv6**.

IPv6 îndeplineşte obiectivele destul de bine. El menţine caracteristicile bune ale IP-ului, le elimină sau atenuează pe cele rele şi adaugă unele noi acolo unde este nevoie. În general, IPv6 nu este compatibil cu IPv4, dar el este compatibil cu celelalte protocoale Internet auxiliare, incluzând TCP, UDP, ICMP, IGMP, OSPF, BGP şi DNS, câteodată fiind necesare mici modificări (majoritatea pentru a putea lucra cu adrese mai lungi). Principalele trăsături ale IPv6 sunt discutate mai jos. Mai multe informaţii despre el pot fi găsite în RFC 2460 până la RFC 2466.

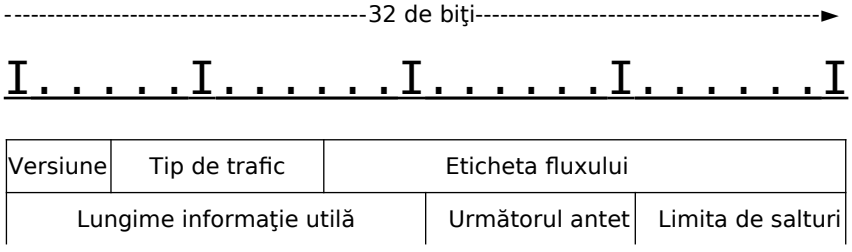
În primul rând şi cel mai important, IPv6 are adrese mai lungi decât IPv4. Ele au o lungime de 16 octeţi, ceea ce rezolvă problema pentru a cărei soluţionare a fost creat IPv6: să furnizeze o sursă efectiv nelimitată de adrese Internet. În curând vom spune mai multe despre adrese.

A doua mare îmbunătăţire a lui IPv6 este simplificarea antetului. El conţine numai 7 câmpuri (faţă de 13 în IPv4). Această schimbare permite ruterelor să prelucreze pachetele mai rapid, îmbunătăţind astfel productivitatea şi întârzierea. De asemenea, vom discuta în curând şi antetul.

A treia mare îmbunătăţire a fost suportul mai bun pentru opţiuni. Această schimbare a fost esenţială în noul antet, deoarece câmpurile care erau necesare anterior sunt acum opţionale. În plus, modul în care sunt reprezentate opţiunile este diferit, uşurând ruterelor saltul peste opţiunile care nu le sunt destinate. Această caracteristică accelerează timpul de prelucrare a pachetelor.

Un al patrulea domeniu în care IPv6 reprezintă un mare progres este în securitate. IETF a avut porţia sa de poveşti de ziar despre copii precoce de 12 ani care îşi folosesc calculatoarele personale pentru a sparge bănci sau baze militare în tot Internet-ul. A existat un sentiment puternic că ar trebui făcut ceva pentru a îmbunătăţi securitatea. Autentificarea şi confidenţialitatea sunt trăsături cheie ale noului IP. Ulterior ele au fost adaptate şi în IPv4, astfel că în domeniul securităţii diferenţele nu mai sunt aşa de mari.

În final, a fost acordată o mai mare atenţie calităţii serviciilor. În trecut s-au făcut eforturi, fără prea mare tragere de inimă, dar acum, o dată cu creşterea utilizării multimedia în Internet, presiunea este şi mai mare.



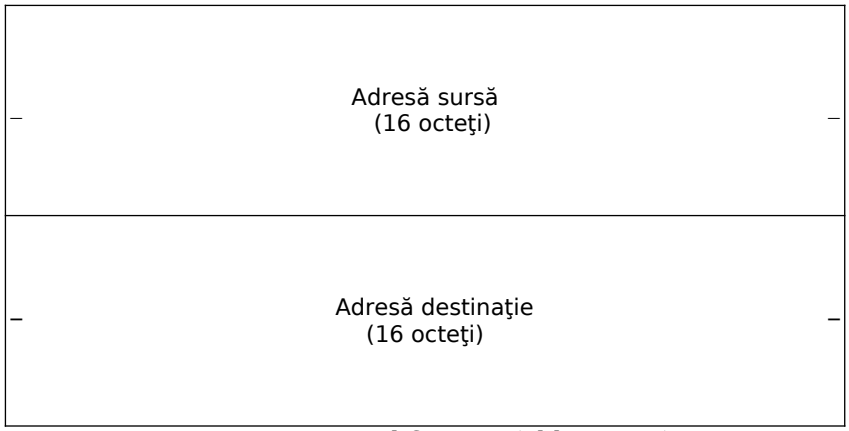


Fig. 5-68. Antetul fix IPv6 (obligatoriu).

Antetul principal IPv6

Antetul IPv6 este prezentat în fig. 5-68. Câmpul *Versiune* este întotdeauna 6 pentru IPv6 (şi 4 pentru IPv4). În timpul perioadei de tranziţie de la IPv4, care va lua probabil un deceniu, ruterele vor fi capabile să examineze acest câmp pentru a spune ce tip de pachet analizează. Ca un efect lateral, acest test iroseşte câteva instrucţiuni pe drumul critic, aşa încât multe implementări vor încerca să-l evite prin folosirea unui câmp din antetul legăturii de date ca să diferenţieze pachetele IPv4 de pachetele IPv6. În acest mod, pachetele pot fi transmise direct rutinei de tratare de nivel reţea corecte. Cu toate acestea, necesitatea ca nivelul legătură de date să cunoască tipurile pachetelor nivelului reţea contravine complet principiul de proiectare care spune că fiecare nivel nu trebuie să cunoască semnificaţia biţilor care îi sunt daţi de către nivelul de deasupra. Discuţiile dintre taberele „Fă-o corect” şi „Fă-o repede” vor fi, fără îndoială, lungi şi virulente.

Câmpul *Tip de trafic (Traffic class)* este folosit pentru a distinge între pachetele care au diverse cerinţe de livrare în timp real. Un câmp cu acest scop a existat în IP de la început, dar a fost implementat sporadic de către rutere. În acest moment se desfăşoară experimente pentru a determina cum poate fi utilizat cel mai bine pentru transmisii multimedia.

Câmpul *Eticheta fluxului* este încă experimental, dar va fi folosit pentru a permite unei surse şi unei destinaţii să stabilească o pseudo-conexiune cu proprietăţi şi cerinţe particulare. De exemplu, un şir de pachete de la un proces de pe o anumită gazdă sursă către un anumit proces pe o anumită gazdă destinaţie poate avea cerinţe de întârziere stricte şi din acest motiv necesită capacitate de transmisie rezervată. Fluxul poate fi stabilit în avans şi poate primi un identificator. Când apare un pachet cu o *Etichetă a fluxului* diferită de zero, toate ruterele pot să o caute în tabelele interne pentru a vedea ce tip de tratament special necesită. Ca efect, fluxurile sunt o încercare de a combina două moduri: flexibilitatea unei subreţele cu datagrame şi garanţiile unei subreţele cu circuite virtuale.

Fiecare flux este desemnat de adresa sursă, adresa destinaţie şi numărul de flux, aşa încât, între o pereche dată de adrese IP pot exista mai multe fluxuri active în acelaşi timp . De asemenea, în acest mod, chiar dacă două fluxuri venind de la

gazde diferite, dar cu acelaşi număr de flux trec prin acelaşi ruter, ruterul va fi capabil să le separe folosind adresele sursă şi destinaţie. Se aşteaptă ca numerele de flux să fie alese aleator, în loc de a fi atribuite secvenţial începând cu 1, pentru că se aşteaptă ca ruterele să le folosească în tabele de dispersie.

Câmpul *Lungimea informaţiei utile* spune câţi octeţi urmează după antetul de 40 de octeţi din fig. 5-68. Numele a fost schimbat faţă de câmpul *Lungime totală* din IPv4 deoarece semnificaţia este uşor modificată: cei 40 de octeţi nu mai sunt parte a lungimii (aşa cum erau înainte).

Câmpul *Antetul următor* dă de gol proiectanţii. Motivul pentru care antetul a putut fi simplificat este că există antete de extensie suplimentare (opţionale). Acest câmp spune care din cele şase antete (actuale) de extensie, dacă există vreunul, urmează după cel curent. Dacă acest antet este ultimul antet IP, câmpul *Antetul următor* spune cărui tip de protocol (de exemplu TCP, UDP) i se va transmite pachetul.

Câmpul *Limita salturilor* este folosit pentru a împiedica pachetele să trăiască veşnic. El este, în practică, identic cu câmpul *Timp de viaţă* din IPv4, şi anume un câmp care este decrementat la fiecare salt dintr-o reţea în alta. În teorie, în IPv4 era un timp în secunde, dar nici un ruter nu-l folosea în acest mod, aşa încât numele a fost modificat pentru a reflecta modul în care este de fapt folosit.

Apoi urmează câmpurile *Adresă sursă* şi *Adresă destinaţie*. Propunerea originală a lui Deering, SIP, folosea adrese de 8 octeţi, dar în timpul procesului de evaluare, multe persoane au simţit că adresele de 8 octeţi s-ar putea epuiza în câteva decenii, în timp ce adresele de 16 octeţi nu s-ar epuiza niciodată. Alte persoane au argumentat că 16 octeţi ar fi un exces, în timp ce alţii încurajau folosirea adreselor de 20 de octeţi pentru a fi compatibile cu protocolul datagramă OSI. O altă grupare dorea adrese de dimensiune variabilă. După multe discuţii, s-a decis că adresele cu lungime fixă de 16 octeţi sunt cel mai bun compromis.

Pentru scrierea adreselor de 16 octeţi a fost inventată o nouă notaţie. Ele sunt scrise ca opt grupuri de câte patru cifre hexazecimale cu semnul : (două puncte) între grupuri, astfel:

8000:0000:0000:0000:0123:4567:89AB:CDEF

Din moment ce multe adrese vor avea multe zerouri în interiorul lor, au fost autorizate trei optimizări. Mai întâi, zerourile de la începutul unui grup pot fi omise, astfel încât 0123 poate fi scris ca 123. În al doilea rând, unul sau mai multe grupuri de 16 zerouri pot fi înlocuite de o pereche de semne două puncte (:). Astfel, adresa de mai sus devine acum

8000::123:4567:89AB:CDEF

În final, adresele IPv4 pot fi scrise ca o pereche de semne două puncte şi un număr zecimal în vechea formă cu punct, de exemplu

::192.31.20.46

Probabil că nu este necesar să fim atât de expliciţi asupra acestui lucru, dar există o mulţime de adrese de 16 octeţi. Mai exact, sunt 2 adrese, care reprezintă aproximativ 3x10 . Dacă întreaga planetă, pământ şi apă, ar fi acoperite cu calculatoare, IPv6 ar permite $7x10^{23}$ adrese IP pe metru pătrat. Studenţii de la

chimie vor observa că acest număr este mai mare decât numărul lui Avogadro. Deşi nu a existat intenţia de a da fiecărei molecule de pe suprafaţa planetei adresa ei IP, nu suntem chiar aşa de departe de aceasta.

În practică, spaţiul de adrese nu va fi folosit eficient, aşa cum nu este folosit spaţiul de adrese al numerelor de telefon (prefixul pentru Manhattan, 212, este aproape plin, dar cel pentru Wyoming, 307, este aproape gol). În RFC 3194, Durand şi Huitema a calculat că, folosind ca referinţă alocarea numerelor de telefon, chiar şi în cel mai pesimist scenariu, vor fi totuşi mult peste 1000 de adrese IP pe metru pătrat de suprafaţă planetară (pământ sau apă). În orice scenariu credibil, vor fi trilioane de adrese pe metru pătrat. Pe scurt, pare improbabil că vom epuiza adresele în viitorul previzibil.

Este instructiv să comparăm antetul IPv4 (fig. 5-53) cu antetul IPv6 (fig. 5-68) pentru a vedea ce a fost eliminat în IPv6. Câmpul *IHL* a dispărut pentru că antetul IPv6 are o lungime fixă. Câmpul *Protocol* a fost scos pentru că în câmpul *Antetul următor* se indică ce urmează după ultimul antet IP (de exemplu, un segment TCP sau UDP).

Toate câmpurile referitoare la fragmentare au fost eliminate, deoarece IPv6 are o abordare diferită a fragmentării. Pentru început, toate gazdele şi ruterele care sunt conforme cu IPv6 trebuie să determine dinamic mărimea datagramei care va fi folosită. Această regulă face ca, de la început, fragmentarea să fie mai puţin probabilă. De asemenea minimul a fost mărit de la 576 la 1280 pentru a permite date de 1024 de octeţi şi mai multe antete. În plus, când o gazdă trimite un pachet IPv6 care este prea mare, ruterul care este incapabil să îl retransmită trimite înapoi un mesaj de eroare în loc să fragmenteze pachetul. Acest mesaj de eroare îi spune gazdei să spargă toate pachetele viitoare către acea destinaţie. Este mult mai eficient să obligi gazdele să trimită de la bun început pachete corecte dimensional, decât să obligi ruterele să le fragmenteze din mers.

În sfârşit, câmpul *Sumă de control* este eliminat deoarece calculul acesteia reduce mult performanţele. Datorită reţelelor fiabile folosite acum, combinate cu faptul că nivelurile de legătură de date şi de transport au în mod normal propriile sume de control, valoarea a încă unei sume de control nu merita preţul de performanţă cerut. Eliminarea tuturor acestor caracteristici a avut ca rezultat un protocol de nivel reţea simplu şi eficient. Astfel, obiectivul lui IPv6 – un protocol rapid, dar flexibil, cu o bogăţie de spaţiu de adrese – a fost atins prin acest proiect.

Antete de extensie

Câteva din câmpurile care lipsesc sunt încă necesare ocazional, astfel încât IPv6 a introdus conceptul de **antet de extensie** (opţional). Aceste antete pot fi furnizate pentru a oferi informaţii suplimentare, dar codificate într-un mod eficient. În prezent sunt definite şase tipuri de antete de extensie, prezentate în fig. 5-69. Fiecare este opţional, dar dacă sunt prezente mai multe, ele trebuie să apară imediat după antetul fix şi, preferabil, în ordinea prezentată.

Antet de extensie	Descriere
Opţiuni salt-după-salt	Diverse informaţii pentru rutere
Opţiuni pentru destinaţie	Informaţii suplimentare pentru
Dirijare	Calea, parţială sau totală, de urmat
Fragmentare	Gestiunea fragmentelor

Autentificare	Verificarea identităţii emiţătorului
Informaţie de siguranţă	Informaţii despre conţinutul criptat

Fig. 5-69. Antetele de extensie IPv6.

Unele dintre antete au un format fix; altele conţin un număr variabil de câmpuri de lungime variabilă. Pentru acestea, fiecare element este codificat ca un tuplu (Tip, Lungime, Valoare). *Tipul* este un câmp de 1 octet care spune ce opţiune este aceasta. Valorile *Tipului* au fost alese astfel, încât primii 2 biţi spun ruterelor care nu ştiu cum să proceseze opţiunea ce anume să facă. Variantele sunt: sărirea opţiunii; eliminarea pachetului; eliminarea pachetului şi trimiterea înapoi a unui pachet ICMP; la fel ca mai înainte, doar că nu se trimit pachete ICMP pentru adrese de trimitere multiplă (pentru a preveni ca un pachet eronat de multicast să genereze milioane de răspunsuri ICMP).

Câmpul *Lungime* este de asemenea un câmp de lungime 1 octet. El precizează cât de lungă este valoarea (de la 0 la 255). *Valoarea* este orice informaţie necesară, până la 255 octeţi.

Antetul salt-după-salt este folosit pentru informaţii ce trebuie examinate de toate ruterele de pe cale. Până acum a fost definită o opţiune: suportul pentru datagrame ce depăşesc de 64K. Formatul acestui antet este prezentat în fig. 5-70. Atunci când este folosit, câmpul *Lungimea informaţiei utile* din antetul fix este zero.

Antetul următor	0	194	4
Lungimea informaţiei utile foarte mari			

Fig. 5-70. Antetul de extensie salt-după-salt pentru datagrame mari (jumbograme).

Ca toate antetele de extensie, acesta începe cu un octet care spune ce tip de antet este următorul. Acest octet este urmat de unul care spune cât de lung este antetul salt-după-salt în octeţi, excluzând primii 8 octeţi, care sunt obligatorii. Toate extensiile încep la fel.

Următorii 2 octeţi arată că această opţiune defineşte dimensiunea datagramei (codul 194) ca un număr de 4 octeţi. Ultimii 4 octeţi dau dimensiunea datagramei. Dimensiunile mai mici de 65.536 nu sunt permise şi au ca rezultat eliminarea pachetului de către primul ruter, care va trimite înapoi un mesaj ICMP de eroare. Datagramele care folosesc acest antet de extensie sunt numite **jumbograme** . Folosirea jumbogramelor este importantă pentru aplicaţiile supercalculatoarelor care trebuie să transfere gigaocteţi de date eficient prin intermediul Internet-ului.

Antetul opţiunilor pentru destinaţie este prevăzut pentru câmpuri care trebuie interpretate numai de către gazda destinaţie. În versiunea iniţială a IPv6 singura opţiune definită este cea de completare a acestui antet până la un multiplu de 8 octeţi, aşa că pentru început nu va fi folosit. El a fost inclus pentru a asigura posibilitatea ca un nou software al ruterului sau al gazdei să îl poate trata, în cazul în care cineva, cândva, se gândeşte la o opţiune pentru destinaţie.

Antetul de dirijare enumeră unul sau mai multe rutere care trebuie să fie vizitate pe calea spre destinaţie. Este foarte asemănător cu dirijarea aproximativă din IPv4 prin aceea că toate adresele listate trebuie vizitate în ordine, dar între acestea pot fi

vizitate alte rutere nelistate. Formatul antetului de dirijare este prezentat în fig. 5-71.

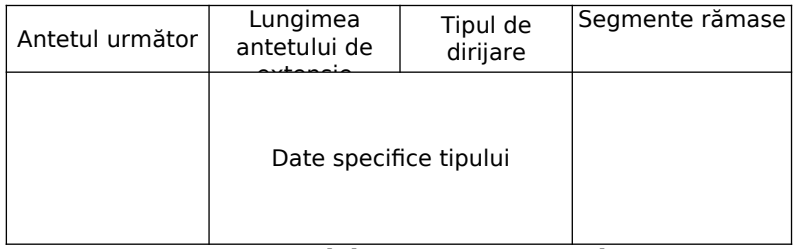


Fig. 5-71. Antetul de extensie pentru dirijare.

Primii 4 octeţi ai antetului de extensie de dirijare conţin patru întregi de 1 octet. Câmpurile *Antet următor* şi *Lungimea antetului* extensie au fost descrise anterior. Câmpul *Tipul dirijării* precizează formatul părţii rămase din header. Tipul 0 arată că după primul cuvânt urmează un cuvânt rezervat pe 32 de biţi, urmat de un număr de adrese IPv6. În viitor, în funcţie de necesităţi, ar putea fi inventate alte tipuri. În final, câmpul *Segmente rămase* reţine câte adrese din listă nu au fost vizitate şi este decrementat de fiecare dată când este vizitată o adresă. Atunci când se ajunge la 0 pachetul este pe cont propriu, fără nici o indicaţie referitoare la ruta de urmat. De obicei, în acest punct este atât de aproape de destinaţie încât calea cea mai bună este evidentă.

Antetul fragment tratează fragmentarea într-un mod similar cu cel al IPv4. Antetul menţine identificatorul datagramei, numărul de fragment şi un bit care spune dacă mai urmează fragmente. În IPv6, spre deosebire de IPv4, numai gazda sursă poate fragmenta un pachet. Ruterele de pe cale nu pot face acest lucru. Deşi această schimbare este o rupere filozofică majoră cu trecutul, ea simplifică munca ruterelor şi permite ca dirijarea să se facă mai rapid. Aşa cum s-a menţionat mai sus, dacă un ruter este confruntat cu un pachet care este prea mare, el elimină pachetul şi trimite un pachet ICMP înapoi la sursă. Această informaţie îi permite gazdei sursă să fragmenteze pachetul în bucăţi mai mici folosind acest antet şi apoi să reîncerce.

Antetul de autentificare oferă un mecanism prin care receptorul unui mesaj poate fi sigur de cel care l-a trimis. Informaţia utilă de siguranţă criptată face posibilă criptarea conţinutului unui pachet, astfel încât doar receptorul căruia îi este destinat poate să-l citească. Pentru a-şi realiza misiunea aceste antete folosesc tehnici criptografice.

Controverse

Dat fiind procesul de proiectare deschisă şi opiniile ferm susţinute ale multora dintre persoanele implicate, nu ar trebui să fie o surpriză că multe din deciziile luate pentru IPv6 au fost foarte controversate. În cele ce urmează vom rezuma câteva dintre acestea. Pentru toate amănuntele tăioase, vezi RFC-urile.

Am menţionat deja disputa legată de lungimea adresei. Rezultatul a fost un compromis: adrese de lungime fixă de 16 octeţi.

O altă dispută s-a dus în jurul lungimii câmpului *Limita salturilor*. O tabără a simţit că limitarea numărului maxim de salturi la 255 (implicită în cazul folosirii unui câmp de 8 biţi) ar fi o greşeală grosolană. Până la urmă, căi de 32 de salturi sunt obişnuite în zilele noastre, iar peste 10 ani pot fi obişnuite căi mult mai lungi.

Aceste persoane au argumentat că folosirea unui spaţiu de adrese enorm a fost clarviziune, dar folosirea unui contor minuscul de salturi a fost miopie. După părerea lor, cel mai mare păcat pe care îl poate comite un informatician este să ofere prea puţini biţi într-un loc.

Răspunsul a fost că pot fi aduse argumente pentru lărgirea fiecărui câmp, conducând la un antet umflat. De asemenea, funcţia câmpului *Limita salturilor* este de a împiedica hoinăreala pachetelor pentru un timp îndelungat şi 65.536 de salturi este mult prea mult. În cele din urmă, pe măsură ce Internet-ul creşte, se vor construi din ce în ce mai multe legături de mare distanţă, făcând posibilă ajungerea dintr-o ţară în alta în cel mult o jumătate de duzină de salturi. Dacă este nevoie de mai mult de 125 de salturi pentru a ajunge de la sursă sau destinaţie la porţile lor internaţionale, ceva este în neregulă cu coloanele vertebrale naţionale. Adepţii celor 8 biţi au câştigat această luptă.

O altă problemă spinoasă a fost dimensiunea maximă a pachetului. Comunitatea supercalculatoa- relor a dorit pachete mai mari de 64 KB. Când un supercalculator începe să transfere, aceasta înseamnă într-adevăr lucru serios şi nu doreşte să fie întrerupt după fiecare 64KB. Argumentul împotriva pachetelor mari este că dacă un pachet de 1 MB ajunge la o linie T1 de 1.5 Mbps, acel pachet va monopoliza linia pentru mai mult de 5 secunde, producând o întârziere semnificativă pentru utilizatorii interactivi care partajează linia. Aici s-a ajuns la un compromis: pachetele normale au fost limitate la 64 KB, dar antetul de extensie salt-după-salt poate fi folosit pentru a permite jumbograme.

Un al treilea punct fierbinte a fost eliminarea sumei de control IPv4. Unele persoane au asimilat această mutare cu eliminarea frânelor de la maşină. Făcând acest lucru, maşina devine mai uşoară, astfel încât poate merge mai repede, dar dacă intervine ceva neaşteptat, apar probleme.

Argumentul împotriva sumei de control a fost că orice aplicaţie care are într-adevăr grijă de integritatea datelor trebuie oricum să aibă o sumă de control la nivelul transport, aşa încât menţinerea a încă o sumă în IP (în plus faţă de suma de control a nivelului legătură de date) este un exces. Mai mult, experienţa a arătat că în IPv4 calculul sumei de control IP era foarte costisitoare. Tabăra împotriva sumei de control a învins de această dată, deci IPv6 nu are o sumă de control.

Gazdele mobile au fost de asemenea un punct de conflict. Dacă un calculator portabil face jumătate din ocolul lumii, poate continua el să opereze la destinaţie cu aceeaşi adresă IPv6 sau trebuie să folosească o schemă cu agenţi locali şi agenţi pentru străini? De asemenea, gazdele mobile introduc asimetrii în sistemul de dirijare. Se poate foarte bine întâmpla ca un mic calculator mobil să audă semnalul puternic trimis de un ruter staţionar, dar ruterul staţionar nu poate auzi semnalul slab trimis de gazda mobilă. În consecinţă, unele persoane au dorit să includă în IPv6 suport explicit pentru gazde mobile. Acest efort a eşuat pentru că nu s-a putut ajunge la un consens pentru o propunere concretă.

Probabil că cea mai mare bătălie a fost pentru securitate. Toată lumea a fost de acord că este necesară. Războiul a fost pentru unde şi cum. Mai întâi unde. Argumentul pentru plasarea la nivelul reţea este că devine un serviciu standard pe care toate aplicaţiile îl pot folosi fără o planificare prealabilă. Argumentul contra este că, în general, aplicaţiile cu adevărat sigure doresc cel puţin criptare capăt-la-

capăt, în care aplicaţia sursă criptează şi aplicaţia destinaţie decriptează. Altfel, utilizatorul este la mila unor implementări potenţial pline de pene a nivelurilor reţea, implementări asupra cărora nu are nici un control. Răspunsul la acest argument este că aceste aplicaţii pot să se abţină de la folosirea facilităţilor de securitate IP şi să-şi facă treaba ele însele. Replica la aceasta este că persoanele care nu au încredere că reţeaua face treaba cum trebuie, nu doresc să plătească preţul unor implementări de IP lente, greoaie, care au această facilitate, chiar dacă este dezactivată.

Un alt aspect al disputei unde să fie pusă securitatea este legat de faptul că multe ţări (dar nu toate) au legi de export severe referitoare la criptografie. Unele, notabil în Franţa şi Irak, reduc în mare măsură folosirea internă a criptografiei, aşa încât oamenii nu pot avea secrete faţă de poliţie. Ca rezultat, orice implementare de IP care utilizează un sistem criptografic suficient de puternic pentru a fi de mare valoare nu poate fi exportat din Statele Unite (şi multe alte ţări) clienţilor din lumea întreagă. Necesitatea menţinerii a două seturi de programe, unul pentru uz intern şi altul pentru export, este un fapt ce întâmpină o opoziţie viguroasă din partea firmelor de calculatoare.

Un punct asupra căruia nu au existat controverse este că nimeni nu se aşteaptă ca Internet-ul bazat pe IPv4 să fie închis într-o duminică dimineaţa şi să repornească ca un Internet bazat pe IPv6 luni dimineaţă. În schimb, vor fi convertite „insule” izolate de IPv6, iniţial comunicând prin tunele. Pe măsură ce insulele IPv6 cresc, ele vor fuziona în insule mai mari. Până la urmă, toate insulele vor fuziona şi Internet-ul va fi convertit complet. Dată fiind investiţia masivă în rutere IPv4 în folosinţă curentă, procesul de conversie va dura probabil un deceniu. Din acest motiv s-a depus o enormă cantitate de efort pentru a asigura că această tranziţie va fi cât mai puţin dureroasă posibil. Pentru mai multe informaţii despre IPv6, vezi (Loshin, 1999).

5.7 REZUMAT

Nivelul reţea furnizează servicii nivelului transport. El se poate baza fie pe circuite virtuale, fie pe datagrame. În ambele cazuri, principala sa sarcină este dirijarea pachetelor de la sursă la destinaţie. În subreţelele bazate pe circuite virtuale, decizia de dirijare se ia atunci când este stabilit circuitul. În subreţelele bazate pe datagrame decizia este luată pentru fiecare pachet.

În reţelele de calculatoare sunt folosiţi mulţi algoritmi de dirijare. Algoritmii statici includ dirijarea pe calea cea mai scurtă şi inundarea. Algoritmii dinamici include dirijarea după vectorul distanţelor şi dirijarea după starea legăturii. Majoritatea reţelelor actuale folosesc unul dintre aceşti algoritmi. Alte teme importante referitoare la dirijare sunt dirijarea ierarhică, dirijarea pentru gazde mobile, dirijarea pentru difuzare, dirijarea multidestinaţie şi cea în reţele punct-la-punct.

Subreţelele pot deveni cu uşurinţă congestionate, mărind întârzierea şi micşorând productivitatea pentru pachete. Proiectanţii reţelelor încearcă să evite congestia printr-o proiectare adecvată. Tehnicile includ politica de retransmitere, folosirea memoriei ascunse, controlul fluxului şi altele. Dacă apare congestia, ea trebuie să fie tratată. Pot fi trimise înapoi pachete de şoc, încărcarea poate fi eliminată sau se pot aplica alte metode.

Următorul pas dincolo de simpla tratare a congestiei este încercarea de a se ajunge la calitatea promisă a serviciului. Metodele care pot fi folosite pentru aceasta includ folosirea zonelor tampon la client, modelarea traficului, rezervarea resurselor şi controlul accesului. Abordările care au fost proiectate pentru o bună calitate a serviciului includ servicii integrate (ca RSVP), servicii diferenţiate şi MPLS.

Reţelele diferă prin multe caracteristici, aşa că atunci când se conectează mai multe reţele, pot să apară probleme. Uneori problemele pot fi evitate prin trecerea prin tunel a unui pachet ce traversează o reţea ostilă, dar dacă reţeaua sursă şi cea destinaţie diferă, această abordare eşuează. Atunci când reţele diferite au dimensiunile maxime ale pachetelor diferite, se poate produce fragmentarea.

Internet-ul posedă o mare varietate de protocoale legate de nivelul reţea. Acestea includ protocolul de transport al datelor, IP, dar şi protocoalele de control ICMP, ARP şi RARP şi protocoalele de dirijare OSPF şi BGP. Internet-ul va rămâne foarte repede fără adrese IP, aşa că s-a dezvoltat o nouă versiune de IP, IPv6.

5.8 PROBLEME

1. Daţi două exemple de aplicaţii pentru care este adecvat un serviciu orientat pe conexiune. Apoi daţi două exemple pentru care un serviciu fără conexiuni este cel mai potrivit.

2. Există vreo situaţie în care un serviciu cu circuit virtual va (sau cel puţin ar putea) livra pachetele în altă ordine? Explicaţi.

3. Subreţelele bazate pe datagrame dirijează fiecare pachet ca pe o unitate separată, independentă de toate celelalte. Subreţelele bazate pe circuite virtuale nu trebuie să facă acest lucru, pentru că fiecare pachet de date urmează o cale predeterminată. Oare această observaţie înseamnă că subreţelele bazate pe circuite virtuale nu au nevoie de capacitatea de a dirija pachetele izolate de la o sursă arbitrară către o destinaţie arbitrară? Explicaţi răspunsul dat.

4. Daţi trei exemple de parametri ai protocolului care ar putea fi negociaţi atunci când este iniţiată o conexiune.

5. Consideraţi următoarea problemă de proiectare, privind implementarea unui serviciu cu circuit virtual. Dacă în interiorul unei subreţele sunt folosite circuite virtuale, fiecare pachet de date trebuie să conţină un antet de 3 octeţi, iar fiecare ruter trebuie să aloce 8 octeţi de memorie pentru identificarea circuitelor. Dacă intern sunt folosite datagrame, sunt necesare antete de 15 octeţi, dar nu este nevoie de spaţiu pentru tabela ruterului. Capacitatea de transmisie costă 1 cent per 10^6 octeţi, per salt. Memoria foarte rapidă pentru ruter poate fi cumpărată la preţul de 1 cent per octet şi se depreciază peste doi ani (considerând numai orele de funcţionare). Din punct de vedere statistic, o sesiune medie durează 1000 de secunde, iar în acest timp sunt transmise 200 de pachete. Un pachet mediu are nevoie de patru salturi. Care implementare este mai ieftină şi cu cât?

6. Presupunând că toate ruterele şi gazdele funcţionează normal şi că întregul software din rutere şi gazde nu conţine nici o eroare, există vreo şansă, oricât de mică, ca un pachet să fie livrat unei destinaţii greşite?

7. Consideraţi reţeaua din fig. 5-7, dar ignoraţi ponderile de pe linii. Presupuneţi că algoritmul de rutare utilizat este cel de inundare. Listaţi toate rutele pe care le va parcurge un pachet trimis de la *A* la *D,* al cărui număr maxim de salturi este 3. De asemenea precizaţi câte noduri consumă inutil bandă de transmisie.

8. Formulaţi o euristică simplă pentru găsirea a două căi de la o sursă dată la o destinaţie
dată care pot supravieţui pierderii oricărei linii de comunicaţie (presupunând că există două astfel de căi). Ruterele sunt considerate suficient de fiabile, deci nu este necesar să ne îngrijoreze posibilitatea căderii ruterelor.

9. Consideraţi subreţeaua din fig. 5-13(a). Se foloseşte dirijarea după vectorul distanţelor şi următorii vectori tocmai au sosit la ruterul C: de la B: (5, 0, 8, 12, 6, 2); de la D: (16, 12, 6, 0, 9, 10); şi de la E: (7, 6, 3, 9, 0, 4). Întârzierile măsurate către B, D şi E, sunt 6, 3 şi respectiv 5. Care este noua tabelă de dirijare a lui C? Precizaţi atât linia de ieşire folosită, cât şi întârzierea presupusă.

10. Dacă întârzierile sunt înregistrate ca numere de 8 octeţi într-o reţea cu 50 de rutere şi vectorii cu întârzieri sunt schimbaţi de două ori pe secundă, cât din lărgimea de bandă a unei linii (duplex integral) este consumată de algoritmul distribuit de dirijare? Presupuneţi că fiecare ruter are trei linii către alte rutere.

11. În fig. 5-14 rezultatul operaţiei SAU logic a celor două mulţimi de biţi ACF este 111 în fiecare linie. Este acesta doar un accident întâmplat aici sau este valabil pentru toate subreţelele, în toate împrejurările?

12. La dirijarea ierarhică cu 4800 de rutere, ce dimensiuni ar trebui alese pentru regiune şi grup, astfel încât să se minimizeze dimensiunea tabelei de dirijare pentru o ierarhie cu trei niveluri? Un punct de pornire este ipoteza că o soluţie cu k clustere de k regiuni de k rutere este aproape de optim, ceea ce înseamnă că valoarea k este aproximativ rădăcina cubică a lui 4800 (aproximativ 16). Folosiţi încercări repetate pentru a verifica combinaţiile cu toţi cei trei parametri în vecinătatea lui 16.

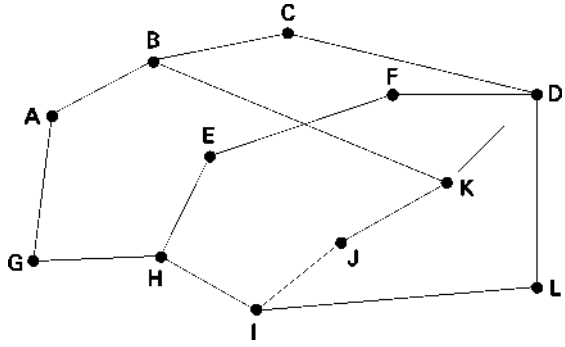
13. În text s-a afirmat că atunci când un sistem gazdă mobil nu este acasă, pachetele trimise către LAN-ul de domiciliu sunt interceptate de agentul său local. Pentru o reţea IP pe un LAN 802.3, cum va realiza agentul local această interceptare?

14. Privind subreţeaua din fig. 5-6, câte pachete sunt generate de o difuzare de la B, folosind:

a) urmărirea căii inverse?
b) arborele de scufundare?

15. Fie reţeaua din fig. 5-16(a). Să ne imaginăm că între F şi G este adăugată o nouă linie, dar arborele de scufundare din fig. 5-16(b) rămâne neschimbat. Ce modificări survin în fig. 5- 16(c) ?

16. Calculaţi un arbore de acoperire pentru trimitere multiplă pentru ruterul C din reţeaua de mai jos pentru un grup cu membrii la ruterele A, B, C, D, E, F, I şi K.



17. În fig. 5-20, difuzează vreodată nodurile H şi I la căutarea pornită din A ?

18. Să presupunem că nodul B din fig. 5-20 tocmai a pornit şi nu are nici o informaţie de dirijare în tabelele sale. Brusc, are nevoie de o cale către H. El va

difuza pachete cu *TTL* setat la 1, 2, 3 şi aşa mai departe. De câte runde are nevoie pentru a găsi o cale ?

19. În cea mai simplă variantă a algoritmului Chord pentru căutarea punct-la-punct, căutările nu folosesc tabela de indicatori. În loc de aceasta, ele sunt lineare în jurul cercului în oricare direcţie. Poate un nod determina cu precizie în ce direcţie trebuie să caute ? Discutaţi răspunsul.

20. Fie cercul Chord din fig. 5-24. Să presupunem că nodul 10 porneşte brusc. Afectează aceasta tabela de indicatori a nodului 1, şi dacă da, cum ?

21. Ca un posibil mecanism de control al congestiei într-o subreţea ce foloseşte intern circuite virtuale, un ruter poate amâna confirmarea unui pachet primit până când (1) ştie că ultima sa transmisie de-a lungul circuitului virtual a fost primită cu succes şi (2) are un tampon liber. Pentru simplitate, să presupunem că ruterele utilizează un protocol stop-and-wait (pas-cu-pas) şi că fiecare circuit virtual are un tampon dedicat pentru fiecare direcţie a traficului. Dacă este nevoie de T sec pentru a trimite un pachet (date sau confirmare) şi sunt n rutere de-a lungul căii, care este viteza cu care pachetele sunt livrate gazdei destinaţie? Presupunem că erorile de transmisie sunt rare, iar conexiunea gazdă-ruter este infinit de rapidă.

22. O subreţea de tip datagramă permite ruterelor să elimine pachete de câte ori este necesar. Probabilitatea ca un ruter să renunţe la un pachet este p. Considerăm cazul unei gazde sursă conectate cu un ruter sursă, care este conectat cu un ruter destinaţie şi apoi cu gazda destinaţie. Dacă unul dintre rutere elimină un pachet, până la urmă gazda sursă va depăşi limita de timp şi va încerca din nou. Dacă liniile gazdă-ruter şi ruter-ruter sunt ambele numărate ca salturi, care este numărul mediu de:

a) salturi per transmisie pe care le face un pachet?
b) transmisii determinate de un pachet?
c) salturi necesare pentru un pachet primit?

23. Descrieţi două diferenţe majore dintre metoda bitului de avertizare şi metoda RED.

24. Daţi o explicaţie pentru faptul că algoritmul găleţii găurite permite un singur pachet per tact, indiferent de cât de mare este pachetul.

25. Într-un sistem oarecare este utilizată varianta cu numărarea octeţilor a algoritmului găleţii găurite. Regula este că pot fi trimise la fiecare tact un pachet de 1024 de octeţi, două pachete de 512 octeţi etc. Formulaţi o limitare serioasă a acestui sistem care nu a fost menţionată în text.

26. O reţea ATM utilizează pentru modelarea traficului o schemă de tip găleată cu jetoane (token bucket). La fiecare 5 psec în găleată este introdus un nou jeton. Fiecare jeton este asociat unei singure celule, care conţine 48 octeţi de date. Care este viteza maximă a datelor care poate fi asigurată?

27. Un calculator dintr-o reţea de 6 Mbps este guvernat de o schemă de tip găleată cu jetoane. Aceasta se umple cu viteza de 1 Mbps. Ea este umplută iniţial la capacitatea maximă, cu 8 megabiţi. Cât timp poate calculatorul să transmită cu întreaga viteză de 6 Mbps?

28. Să ne imaginăm o specificaţie de flux care are dimensiunea maximă a pachetului de 1000 de octeţi, viteza găleţii cu jetoane de 10 milioane de octeţi/sec, capacitatea găleţii de 1 milion de octeţi şi viteza maximă de transmisie de 50 de milioane de octeţi/sec. Cât timp poate dura o rafală la viteza maximă?

29. Reţeaua din fig. 5-37 foloseşte RSVP cu arbori multidestinaţie pentru gazdele 1 şi 2, după cum este ilustrat. Să presupunem că gazda 3 cere un canal cu lăţimea de bandă de 2MB/sec pentru un flux de la gazda 1 şi alt canal cu lărgimea de bandă de 1MB/sec pentru un flux de la gazda 2. În acelaşi timp gazda 4 cere un canal cu lărgimea de bandă de 2MB/sec pentru un flux de la gazda 1 şi gazda 5 cere un alt cu lărgimea de bandă de 1MB/sec pentru un flux de la gazda 2. Ce lărgime de bandă totală va fi rezervată la ruterele A, B, C, E, H, J, K, L pentru aceste cereri?

30. Procesorul dintr-un ruter poate prelucra 2 milioane de pachete/sec. Încărcarea oferită lui este de 1,5 milioane pachete/sec. Dacă o rută de la sursă la destinaţie trece prin 10 rutere, cât timp se consumă în aşteptare şi pentru servirea de către procesoare?

31. Fie utilizatorul unor servicii diferenţiate cu rutare expeditivă. Există o garanţie că pachetele prioritare vor suferi o întârziere mai mică decât pachetele normale? De ce sau de ce nu?

32. Este nevoie de fragmentare în reţele concatenate bazate pe circuite virtuale sau numai în sisteme cu datagrame?

33. Trecerea prin tunel printr-o subreţea de circuite virtuale concatenate este simplă: ruterul multiprotocol de la un capăt stabileşte circuitul virtual către celălalt capăt şi trece pachetele prin el. Poate această trecere prin tunel să fie folosită şi în subreţelele bazate pe datagrame? Dacă da, cum?

34. Să presupunem că gazda *A* este conectată la ruterul *R1, R1* este conectat la alt ruter *R2,* şi *R2* este conectat la gazda *B*. Să presupunem că un mesaj TCP care conţine 900 octeţi de date şi 20 de octeţi de antet TCP este transmis codului IP aflat pe gazda *A* pentru a fi transmis lui *B*. Arătaţi câmpurile *Lungimea totală, Identificare, DF MF* şi *Deplasamentul fragmentului* din antetul IP din fiecare pachet transmis prin cele trei legături. Se presupune că legătura *A-R1* poate suporta o lungime maximă de cadru de 1024 de octeţi incluzând un antet de cadru de 14 octeţi, legătura *R1-R2* poate suporta o lungime maximă de cadru de 512 de octeţi incluzând un antet de cadru de 8 octeţi şi legătura *R2-B* poate suporta o lungime maximă de cadru de 512 octeţi incluzând un antet de cadru de 12 octeţi.

35. Un ruter distruge pachetele IP a căror lungime totală (date plus antet) este de 1024 octeţi. Presupunând că pachetele trăiesc pentru 10 sec, care este viteza maximă a liniei la care poate opera ruterul fără a fi în pericol să cicleze prin spaţiul numerelor de ID al datagramelor IP.

36. O datagramă IP care foloseşte opţiunea *Dirijare strictă de la sursă* trebuie să fie fragmentată. Credeţi că opţiunea este copiată în fiecare fragment, sau este suficient să fie pusă numai în primul fragment? Explicaţi răspunsul.

37. Să presupunem că pentru o adresă de clasă B, partea care specifică reţeaua utilizează 20 de biţi în loc de 16 biţi. Câte reţele de clasă B se pot obţine?

38. Transformaţi adresa IP a cărei reprezentare zecimală este C22F1582 într-o notaţie zecimală cu puncte.

39. O reţea din Internet are masca de subreţea 255.255.240.0. Care este numărul maxim de gazde din subreţea?

40. Un număr mare de adrese IP consecutive sunt disponibile începând cu 198.16.0.0. Să presupunem că patru organizaţii, *A*, *B*, *C*, *D*, cer câte 4000, 2000, 4000 şi 8000 adrese, în această ordine. Precizaţi, pentru fiecare dintre ele, prima şi ultima adresă IP atribuită, precum şi masca în notaţia w.x.y.z/s.

41. Un ruter tocmai a primit următoarele noi adrese IP: 57.6.96.0/21, 57.6.104.0/21, 57.6.112.0/21 şi 57.6.129.0/21. Dacă toate folosesc aceeaşi linie de ieşire, pot fi ele compuse? Dacă da, ce va rezulta? Dacă nu, de ce nu ?

42. Setul de adrese IP de la 29.18.0.0 la 19.18.128.255 au fost reunite la 29.18.9.9/17. Totuşi există un spaţiu de 1024 de adrese nealocate, de la 29.18.60.0 la 29.18.63.255, care sunt brusc alocate unei gazde care foloseşte altă linie de ieşire. Este nevoie acum ca adresa agregată să fie spartă în blocurile constituente, să se adauge blocurile noi la tabelă şi să se vadă apoi dacă este posibilă o altă reunire? Dacă nu, ce se poate face ?

43. Un ruter are următoarele intrări (CIDR) în tabela sa de dirijare :

Adresă/mască	**Următorul salt**
135.46.56.0/22	Interfaţa 0
135.46.60.0/22	Interfaţa 1
192.53.40.0/23	Ruter 1
Implicit	Ruter 2

Pentru fiecare dintre următoarele adrese IP, ce face ruterul dacă primeşte un pachet cu respectiva adresă?

a) 135.46.63.10
b) 135.46.57.14
c) 135.46.52.2
d) 192.53.40.7
e) 192.53.56.7

44. Multe companii au politica de a avea două (sau mai multe) rutere conectate la Internet pentru a avea redundanţă în caz că unul dintre ele nu mai funcţionează. Mai este această politică posibilă cu NAT ? Explicaţi răspunsul.

45. Tocmai i-aţi explicat unui prieten protocolul ARP. Când aţi terminat, el spune: „Am înţeles. ARP oferă un serviciu nivelului reţea, deci face parte din nivelul legăturii de date.” Ce îi veţi spune?

46. Atât ARP cât şi RARP realizează corespondenţa adreselor dintr-un spaţiu în altul. Din acest punct de vedere cele două protocoale sunt similare. Totuşi, implementările lor sunt fundamental diferite. Care este diferenţa esenţială dintre ele?

47. Descrieţi un procedeu pentru reasamblarea fragmentelor IP la destinaţie.

48. Cei mai mulţi algoritmi de reasamblare a datagramelor IP au un ceas pentru a evita ca un fragment pierdut să ţină ocupate pentru totdeauna tampoanele de reasamblare. Să presupunem că o datagramă este împărţită în patru fragmente. Primele trei sosesc, dar ultimul este întârziat. În cele din urmă timpul expiră şi cele trei fragmente sunt eliminate din memoria receptorului. Puţin mai târziu, soseşte şi ultimul fragment. Ce ar trebui făcut cu el?

49. Atât la IP cât şi la ATM, suma de control acoperă numai antetul, nu şi datele. De ce credeţi că s-a ales această soluţie?

50. O persoană care locuieşte în Boston călătoreşte la Minneapolis, luându-şi calculatorul portabil cu sine. Spre surprinderea sa, LAN-ul de la destinaţia din Minneapolis este un LAN IP fără fir, deci nu trebuie să se conecteze. Este oare necesar să se recurgă la întreaga poveste cu agenţi locali şi agenţi străini pentru ca mesajele de poştă electronică şi alte tipuri de trafic să-i parvină corect?

51. IPv6 foloseşte adrese de 16 octeţi. Dacă la fiecare picosecundă este alocat câte un bloc de 1 milion de adrese, cât timp vor exista adrese disponibile?

52. Câmpul *Protocol* folosit în antetul IPv4 nu este prezent în antetul fix pentru IPv6. De ce?

53. Când se introduce protocolul IPv6, protocolul ARP trebuie să fie modificat? Dacă da, modificările sunt conceptuale sau tehnice?

54. Scrieţi un program care să simuleze dirijarea prin inundare. Fiecare pachet ar conţine un contor care este decrementat la fiecare salt. Când contorul ajunge la zero, pachetul este eliminat. Timpul este discret, iar fiecare linie manevrează un pachet într-un interval de timp. Realizaţi trei versiuni ale acestui program: toate liniile sunt inundate, sunt inundate toate liniile cu excepţia liniei de intrare, sau sunt inundate numai cele mai bune k linii (alese statistic). Comparaţi inundarea cu dirijarea deterministă ($k = 1$) în termenii întârzierii şi lărgimii de bandă folosite.

55. Scrieţi un program care simulează o reţea de calculatoare ce foloseşte un timp discret. Primul pachet din coada de aşteptare a fiecărui ruter face un salt per interval de timp. Fiecare ruter are numai un număr finit de zone tampon. Dacă un pachet soseşte şi nu este loc pentru el, el este eliminat şi nu mai este retransmis. În schimb, există un protocol capăt-la- capăt, complet, cu limită de timp şi pachete de confirmare, care va regenera în cele din urmă pachetul de la ruterul sursă. Reprezentaţi grafic productivitatea reţelei ca funcţie de limita de timp, parametrizată de rata erorilor.

56. Scrieţi o funcţie pentru retransmiterea într-un ruter IP. Procedura are un parametru, o adresă IP. De asemenea are acces la o tabelă globală constând dintr-un vector de tripleţi. Fiecare triplet conţine trei întregi: o adresă IP, o mască de subreţea şi linia de ieşire ce trebuie folosită. Funcţia caută adresa IP în tabelă folosind CIDR şi întoarce ca valoare linia ce trebuie folosită.

57. Folosiţi programele *traceroute* (UNIX) sau *tracert* (Windows) pentru a urmări calea de la calculatorul personal până la diverse universităţi de pe alte continente. Creaţi o listă a legăturilor transoceanice pe care le-aţi descoperit. Câteva sit-uri de încercat sunt: www.berkely.edu (California), www.u-tokyo.ac.jp (Tokyo), www.mit.edu (Massachusetts), www.vu.nl (Amsterdam), www.usyd.edu.au (Sydney), www.ucl.ac.uk (Londra), www.uct.ac.za (Cape Town).

NIVELUL TRANSPORT

Nivelul transport nu este doar un alt nivel, el este miezul întregii ierarhii de protocoale. Sarcina sa este de a transporta date de la maşina sursă la maşina destinaţie într-o manieră sigură şi eficace din punctul de vedere al costurilor, independent de reţeaua sau reţelele fizice utilizate. Fără nivelul transport şi-ar pierde sensul întregul concept de ierarhie de protocoale. În acest capitol vom studia în detaliu nivelul transport, incluzând serviciile, arhitectura, protocoalele şi performanţele sale.

6.1 SERVICIILE OFERITE DE NIVELUL TRANS PORT

În secţiunile următoare vom face o prezentare a serviciilor oferite de nivelul transport. Vom studia serviciile oferite nivelului aplicaţie. Pentru a face problema serviciului de transport mai concretă, vom examina două seturi de primitive ale nivelului transport. La început ne vom ocupa de unul simplu (dar ipotetic), pentru a arăta ideile de bază. Apoi va fi studiată interfaţa folosită în mod obişnuit în Internet.

6.1.1 Servicii furnizate nivelurilor superioare

Scopul principal al nivelului transport este de a oferi servicii eficiente, sigure şi ieftine utilizatorilor, în mod normal procese aparţinând nivelului aplicaţie. Pentru a atinge acest scop, nivelul transport utilizează serviciile oferite de nivelul reţea.

Hardware-ul şi/sau software-ul care se ocupă de toa-

te acestea în cadrul nivelului transport poartă numele de **entitate de transport**. Entitatea de transport poate aparţine nucleului sistemului de operare, unui proces distinct, unei biblioteci legate de aplicaţiile de reţea sau poate fi găsită în cadrul plăcii de reţea. Relaţia (logică) între nivelurile reţea, transport şi aplicaţie este prezentată în fig. 6-1.

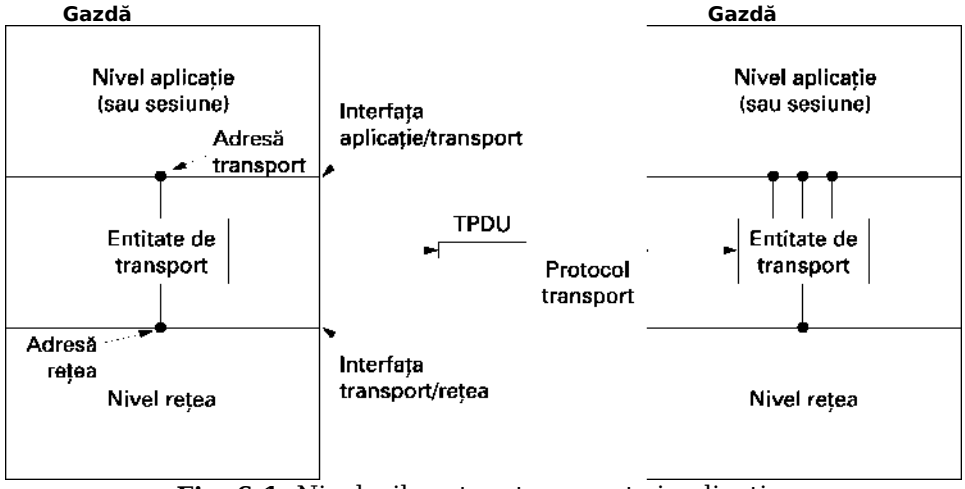


Fig. 6-1. Nivelurile reţea, transport şi aplicaţie.

Cele două tipuri de servicii: orientate pe conexiune sau datagramă, existente în cadrul nivelului reţea, se regăsesc şi la acest nivel. Serviciul orientat pe conexiune de la nivelul transport are multe asemănări cu cel de la nivel reţea. În ambele cazuri, conexiunile au trei faze: stabilirea conexiunii, transferul de date şi eliberarea conexiunii. Adresarea şi controlul fluxului sunt şi ele similare pentru ambele niveluri. Mai mult, chiar şi serviciul fără conexiune al nivelului transport este foarte asemănător cu cel al nivelului reţea.

O întrebare evidentă este atunci: dacă serviciile la nivel transport sunt atât de asemănătoare cu cele de la nivel reţea, de ce este nevoie de două niveluri distincte? De ce nu este suficient un singur nivel? Răspunsul este unul subtil, dar extrem de important, şi ne cere să ne întoarcem la fig. l-9. Codul pentru nivelul transport este executat în întregime pe maşinile utilizatorilor, dar nivelul reţea este executat în cea mai mare parte de mediul de transport (cel puţin pentru reţelele larg răspândite geografic – WAN). Ce s-ar întâmpla dacă nivelul reţea ar oferi servicii neadecvate? Dar dacă acesta ar pierde frecvent pachete? Ce se întâmplă dacă din când în când ruterul cade?

Ei bine, în toate aceste cazuri apar probleme. Deoarece utilizatorii nu pot controla nivelul reţea, ei nu pot rezolva problema unor servicii de proastă calitate folosind rutere mai bune sau adăugând o tratare a erorilor mai sofisticată la nivelul legătură de date. Singura posibilitate este de a pune deasupra nivelului reţea un alt nivel care să amelioreze calitatea serviciilor. Dacă pe o subreţea orientată pe conexiune, o entitate de transport este informată la jumătatea transmisiei că a fost închisă abrupt conexiunea sa la nivel reţea, fără nici o indicaţie despre ceea ce s-a întâmplat cu datele aflate în acel moment în tranzit, ea poate iniţia o altă conexiune

la nivel reţea cu entitatea de transport aflată la distanţă. Folosind această nouă conexiune, ea îşi poate întreba corespondenta care date au ajuns la destinaţie şi care nu, şi poate continua comunicarea din locul de unde a fost întreruptă.

În esenţă, existenţa nivelului transport face posibil ca serviciile de transport să fie mai sigure decât cele echivalente de la nivelul reţea. Pachetele pierdute sau incorecte pot fi detectate şi corectate de către nivelul transport. Mai mult, primitivele serviciului de transport pot fi implementate ca apeluri către procedurile de bibliotecă, astfel încât să fie independente de primitivele de la nivelul reţea. Apelurile nivelului reţea pot să varieze considerabil de la o reţea la alta (de exemplu, serviciile fără conexiune într-o reţea locală pot fi foarte diferite de serviciile orientate pe conexiune dintr-o reţea larg răspândită geografic). Ascunzând serviciul reţea în spatele unui set de primitive ale serviciului transport, schimbarea serviciului reţea necesită numai înlocuirea unui set de proceduri de bibliotecă cu un altul care face acelaşi lucru, cu un serviciu inferior diferit.

Mulţumită nivelului transport, programatorii de aplicaţii pot scrie cod conform unui set standard de primitive, pentru a rula pe o mare varietate de reţele, fără să îşi pună problema interfeţelor de subreţea diferite sau transmisiilor nesigure. Dacă toate reţelele reale ar fi perfecte şi toate ar avea acelaşi set de primitive şi ar fi garantate să nu se schimbe niciodată, atunci probabil că nivelul transport nu ar mai fi fost necesar. Totuşi, în lumea reală el îndeplineşte importanta funcţie de a izola nivelurile superioare de tehnologia, arhitectura şi imperfecţiunile subreţelei.

Din această cauză, în general se poate face o distincţie între nivelurile de la 1 la 4, pe de o parte, şi cel (cele) de deasupra, pe de altă parte. Primele pot fi văzute ca **furnizoare de servicii de transport,** iar ultimele ca **utilizatoare de servicii de transport**. Această distincţie între utilizatori şi furnizori are un impact considerabil în ceea ce priveşte proiectarea arhitecturii de niveluri şi conferă nivelului transport o poziţie cheie, acesta fiind limita între furnizorul şi utilizatorul serviciilor sigure de transmisie de date.

6.1.2 Primitivele serviciilor de transport

Pentru a permite utilizatorului să acceseze serviciile de transport, nivelul transport trebuie să ofere unele operaţii programelor aplicaţie, adică o interfaţă a serviciului transport. Fiecare serviciu de transport are interfaţa sa. În acest capitol, vom examina mai întâi un serviciu de transport simplu (ipotetic) şi interfaţa sa pentru a vedea aspectele esenţiale. În secţiunea următoare vom analiza un exemplu real.

Serviciul transport este similar cu cel reţea, dar există şi câteva diferenţe importante. Principala diferenţă este că serviciul reţea a fost conceput pentru a modela serviciile oferite de reţelele reale. Acestea pot pierde pachete, deci serviciile la nivel reţea sunt în general nesigure.

În schimb, serviciile de transport (orientate pe conexiune) sunt sigure. Desigur, în reţelele reale apar erori, dar este tocmai acesta este scopul nivelului transport: să furnizeze un serviciu sigur deasupra unui nivel reţea nesigur.

Ca exemplu, să considerăm două procese conectate prin ‘pipe'-uri (tuburi) în UNIX. Acestea presupun o conexiune perfectă între ele. Ele nu vor să aibă de-a face cu confirmări, pachete pierdute, congestii sau altele asemănătoare. Ele au nevoie de o conexiune sigură în proporţie de 100%. Procesul A pune datele la un capăt al tubului, iar procesul B le ia de la celălalt capăt. Aceasta este exact ceea ce face un serviciu transport orientat pe conexiune: ascunde imperfecţiunile reţelei, astfel încât procesele utilizator pot să presupună existenţa unui flux de date fără erori.

În acelaşi timp nivelul transport furnizează şi un serviciu nesigur. Totuşi, sunt puţine de spus în legătură cu acesta, aşa că în acest capitol ne vom concentra atenţia asupra serviciului orientat pe conexiune. Cu toate acestea, există unele aplicaţii, cum sunt programele client-server şi fluxurile multimedia, care beneficiază de transport fără conexiune, deci vom vorbi puţin despre ele mai târziu.

O a doua diferenţă între serviciul reţea şi cel de transport se referă la destinaţiile lor. Serviciul reţea este folosit doar de entităţile de transport. Puţini utilizatori scriu ei înşişi entităţile de transport şi, astfel, puţini utilizatori sau programe ajung să vadă vreodată serviciile reţea aşa cum sunt ele. În schimb, multe programe (şi programatori) folosesc primitivele de transport. De aceea, serviciul transport trebuie să fie uşor de utilizat.

Ca să ne facem o idee despre cum poate arăta un serviciu de transport, să considerăm cele cinci primitive prezentate în fig. 6-2. Această interfaţă este într-adevăr simplă, dar prezintă trăsăturile de bază ale oricărei interfeţe orientate pe conexiune a nivelului transport. Ea permite programelor de aplicaţie să stabilească, să utilizeze şi să elibereze conexiuni, ceea ce este suficient pentru multe aplicaţii.

Primitiva	Unitatea de date	Explicaţii
LISTEN	(nimic)	Se blochează până când un proces încearcă să se
CONNECT	CONNECTION REQ.	Încearcă să stabilească conexiunea
SEND	DATE	Transmite informaţie
RECEIVE	(nimic)	Se blochează până când primeşte date trimise
DISCONNECT	DISCONNECTION REQ.	T rimisă de partea care vrea să se deconecteze

Fig. 6-2. Primitivele unui serviciu de transport simplu.

Pentru a vedea cum pot fi utilizate aceste primitive, să considerăm o aplicaţie cu un server şi un număr oarecare de clienţi la distanţă. La început, serverul apelează primitiva LISTEN, în general prin apelul unei funcţii de bibliotecă care face un apel sistem pentru a bloca serverul până la apariţia unei cereri client. Atunci când un client vrea să comunice cu serverul, el va executa un apel CONNECT. Entitatea de transport tratează acest apel blocând apelantul şi trimiţând un pachet la server. Acest pachet încapsulează un mesaj către entitatea de transport de pe server.

Este momentul să facem câteva precizări în legătură cu terminologia. În lipsa unui termen mai bun vom folosi acronimul **TPDU (Transport Protocol Data Unit –** unitate de date a protocolului de transport) pentru toate mesajele schimbate între două entităţi de transport corespondente. Astfel, TPDU-urile (schimbate la nivelul transport) sunt conţinute în pachete (utilizate de nivelul reţea). La rândul lor, pachetele sunt conţinute în cadre (utilizate la nivelul legătură de date). Atunci când este primit un cadru, nivelul legătură de date prelucrează antetul cadrului şi dă

conţinutul util nivelului reţea. Entitatea reţea prelucrează antetul pachetului şi pasează conţinutul util entităţii de transport. Această ierarhie este ilustrată în fig. 6-3.

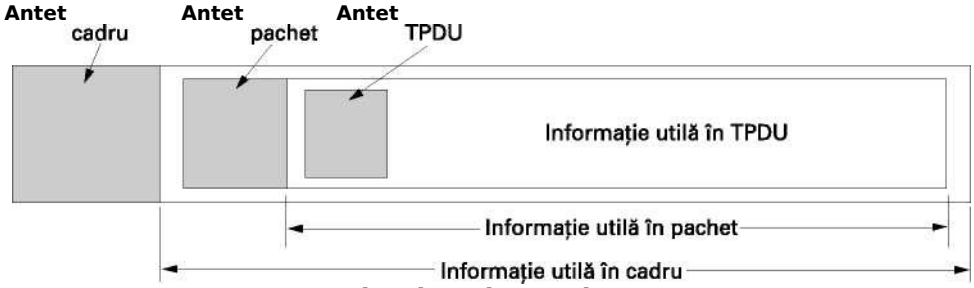


Fig. 6-3. Ierarhia de cadre, pachete şi TPDU-uri.

Revenind la exemplul nostru, apelul CONNECT al clientului generează un TPDU de tip CONNECTION REQUEST care îi este trimis serverului. Atunci când acesta ajunge, entitatea de transport verifică dacă serverul este blocat într-un apel LISTEN (deci dacă aşteaptă o cerere de conexiune). În acest caz, deblochează serverul şi trimite înapoi clientului un TPDU CONNECTION ACCEPTED. Atunci când acest TPDU ajunge la destinaţie, clientul este deblocat şi conexiunea este stabilită.

Acum pot fi schimbate date folosindu-se primitivele SEND şi RECEIVE. Cea mai simplă posibilitate este ca una din părţi să facă un apel RECEIVE (blocant) aşteptând ca cealaltă parte să execute un SEND. Atunci când soseşte un TPDU, receptorul este deblocat. El poate prelucra TPDU-ul şi trimite o replică. Atâta vreme cât amândouă părţile ştiu cine este la rând să trimită mesaje şi cine este la rând să recepţioneze, totul merge bine.

Trebuie să observăm că la nivelul transport, chiar şi un schimb de date simplu, unidirecţional, este mult mai complicat decât la nivelul reţea. Fiecare pachet de date trimis va fi (în cele din urmă) confirmat. Pachetele care conţin TPDU-uri de control sunt de asemenea confirmate, implicit sau explicit. Aceste confirmări sunt gestionate de entităţile de transport folosind protocoalele de la nivelul reţea şi nu sunt vizibile utilizatorilor nivelului transport. Similar, entităţile de transport trebuie să se ocupe de ceasuri şi de retransmisii. Nimic din tot acest mecanism nu este vizibil pentru utilizatorii nivelului transport, pentru care o conexiune este un tub fără pierderi: un utilizator îndeasă biţi la un capăt şi aceştia apar, ca prin minune, la capătul celalalt. Această capacitate de a ascunde complexitatea este motivul care face ierarhia de protocoale să fie un instrument atât de puternic.

TPDU: Cerere

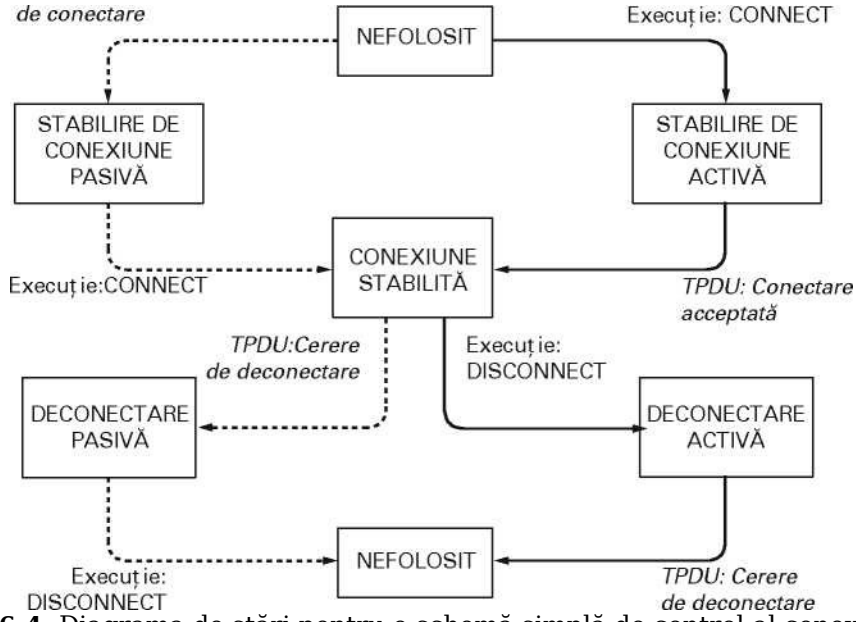


Fig. 6-4. Diagrama de stări pentru o schemă simplă de control al conexiunii. Tranziţiile etichetate cu italice sunt cauzate de sosirea unor pachete. Liniile continue indică secvenţa de stări a clientului. Liniile punctate indică secvenţa de stări a serverului.

Atunci când o conexiune nu mai este necesară, ea trebuie eliberată pentru a putea elibera şi spaţiul alocat în tabelele corespunzătoare din cele două entităţi de transport. Deconectările se pot face în două variante: asimetrică sau simetrică. În varianta asimetrică, oricare dintre utilizatori poate apela o primitivă DISCONNECT, ceea ce va avea ca rezultat trimiterea unui TPDU DISCONNECT REQUEST entităţii de transport aflate la distanţă. La sosirea acestuia conexiunea este eliberată.

În varianta simetrică fiecare direcţie este închisă separat, independent de cealaltă. Atunci când una din părţi face un apel DISCONNECT, însemnând că nu mai sunt date de trimis, ea va putea încă recepţiona datele transmise de entitatea de transfer aflată la distanţă. În acest model conexiunea este eliberată dacă ambele părţi au apelat DISCONNECT.

O diagramă de stări pentru stabilirea şi eliberarea conexiunilor folosind aceste primitive simple este prezentată în fig. 6-4. Fiecare tranziţie este declanşată de un eveniment: fie este executată o primitivă de către utilizatorul local al nivelului transport, fie este primit un pachet. Pentru simplitate vom presupune că fiecare TPDU este confirmat separat. Vom presupune de asemenea că este folosit un model de deconectare simetric, clientul iniţiind acţiunea. Trebuie reţinut că acesta este un model foarte simplu, în secţiunile următoare vom analiza modele reale.

6.1.3 Socluri Berkeley

Vom trece în revistă acum un alt set de primitive de transport: primitivele pentru socluri TCP folosite în sistemul de operare Berkeley-UNIX. Primitivele sunt

enumerate în fig. 6-5. În general putem spune că acestea sunt similare modelului din capitolul precedent, dar oferă mai multe caracteristici şi flexibilitate. Nu vom detalia TPDU-urile existente; această discuţie mai are de aşteptat până în momentul când vom studia TCP, mai târziu, în acest capitol.

Primitiva	Funcţia
SOCKET	Creează un nou punct de capăt al comunicaţiei
BIND	Ataşează o adresă locală la un soclu
LISTEN	Anunţă capacitatea de a accepta conexiuni; determină
ACCEPT	Blochează apelantul până la sosirea unei cereri de conexiune
CONNECT	Tentativă (activă) de a stabili o conexiune
SEND	Trimite date prin conexiune
RECEIVE	Recepţionează date prin conexiune
CLOSE	Eliberează conexiunea

Fig. 6-5. Primitivele pentru socluri TCP

Primele patru primitive din tabel sunt executate, în această ordine, de către server. Primitiva SOCKET creează un nou capăt al conexiunii şi alocă spaţiu pentru el în tabelele entităţii de transport. În parametrii de apel se specifică formatul de adresă utilizat, tipul de serviciu dorit (de exemplu, flux sigur de octeţi) şi protocolul. Un apel SOCKET reuşit întoarce un descriptor de fişier (la fel ca un apel OPEN) care va fi utilizat în apelurile următoare.

Soclurile nou create nu au încă nici o adresă. Ataşarea unei adrese se face utilizând primitiva BIND. Odată ce un server a ataşat o adresă unui soclu, clienţii se pot conecta la el. Motivul pentru care apelul SOCKET nu creează adresa direct este că unor procese le pasă de adresa lor (de exemplu, unele folosesc aceeaşi adresă de ani de zile şi oricine cunoaşte această adresă), în timp ce altele nu.

Urmează apelul LISTEN, care alocă spaţiu pentru a reţine apelurile primite în cazul când mai mulţi clienţi încearcă să se conecteze în acelaşi timp. Spre deosebire de modelul din primul nostru exemplu, aici LISTEN nu mai este un apel blocant.

Pentru a se bloca şi a aştepta un apel, serverul execută o primitivă ACCEPT. Atunci când soseşte un TPDU care cere o conexiune, entitatea de transport creează un nou soclu cu aceleaşi proprietăţi ca cel iniţial şi întoarce un descriptor de fişier pentru acesta. Serverul poate atunci să creeze un nou proces sau fir de execuţie care va gestiona conexiunea de pe noul soclu şi să aştepte în continuare cereri de conexiune pe soclul iniţial. ACCEPT returnează un descriptor normal de fişier, care poate fi folosit pentru citirea şi scrierea în mod standard, la fel ca pentru fişiere.

Să privim acum din punctul de vedere al clientului: şi în acest caz, soclul trebuie creat folosind o primitivă SOCKET, dar primitiva BIND nu mai este necesară, deoarece adresa folosită nu mai este importantă pentru server. Primitiva CONNECT blochează apelantul şi demarează procesul de conectare. Când acesta s-a terminat (adică atunci când TPDU-ul corespunzător a fost primit de la server), procesul client este deblocat şi conexiunea este stabilită. Atât clientul cât şi serverul pot utiliza acum primitivele SEND şi RECEIVE pentru a transmite sau recepţiona date folosind o conexiune duplex integral. Se pot folosi şi apelurile de

sistem READ şi WRITE standard din UNIX, dacă nu sunt necesare opţiunile speciale oferite de SEND şi RECV.

Eliberarea conexiunii este simetrică. Atunci când ambele părţi au executat primitiva CLOSE, conexiunea este eliberată.

6.1.4 Un exemplu de programare cu socluri: server de fişiere pentru Internet

Ca exemplu de cum pot fi folosite apelurile pentru socluri, vom considera codurile client şi server din fig. 6-6. Aici avem un server de Internet foarte primitiv împreună cu un exemplu de client care îl utilizează. Codul are multe limitări (discutate mai jos), dar în principiu codul server poate fi compilat şi rulat pe orice sistem UNIX conectat la Internet. Codul client poate fi apoi compilat şi rulat pe orie altă maşină UNIX din Internet, oriunde în lume. Codul client poate fi executat cu parametrii adecvaţi pentru a obţine orice fişier la care serverul are acces pe maşina sa. Fişierul este scris la ieşirea standard, care, desigur, poate fi redirectată spre un fişier sau spre o conductă (pipe).

Să ne uităm mai întâi la codul server. Acesta începe incluzând nişte „header”-e (antete) standard între care ultimele 3 conţin principalele definiţii şi structuri de date care se referă la Internet. Apoi urmează o definire a *SERVER_PORT* (portului de server) ca 12345. Acest număr a fost ales arbitrar. Orice număr între 1024 şi 65535 va funcţiona la fel de bine atâta timp cât nu este utilizat de alte procese. Bineînţeles, clientul şi serverul trebuie sa folosească acelaşi port. Dacă serverul va deveni vreodată un succes de talie mondială (improbabil, ştiind cât de primitiv este) îi va fi asignat un port permanent sub 1024 şi va apărea la *www.iana.org*.

Următoarele două linii în codul server definesc două constante necesare. Prima determină dimensiunea zonei de memorie folosite pentru transferul de fişiere. A doua determină cât de multe conexiuni în aşteptare pot fi reţinute înainte ca cele care urmează să fie înlăturate după sosire.

După declaraţiile variabilelor locale începe codul server. Acesta porneşte cu iniţializarea unei structuri de date care va ţine adresa IP a serverului. Această structură de date va fi în curând asociată cu soclul serverului. Apelul către *memset* setează structura de date la 0. Cele trei atribuiri care îi urmează completează trei din câmpurile sale. Ultima dintre ele conţine portul serverului. Funcţiile *htonl* şi *htons* se referă la conversia valorilor într-un format standard astfel încât codul să ruleze corect atât pe maşini „big-endian” (de exemplu SPARC) cât şi maşini „little-endian” (de exemplu Pentium). Semantica lor exactă nu este relevantă aici.

```
/* Această pagină conţine un program client care poate cere un fişier de la programul server
de pe pagina următoare. Serverul răspunde trimiţând întregul fişier.
*/ #include <sys/types.h> #include <sys/socket.h> #include <netinet/în.h> #include
<netdb.h>

 #define SERVER_PORT 12345        /* arbitrar, dar clientul şi serverul trebuie să fie de acord */
 #define BUF_SIZE 4096                                  /* dimensiunea blocului de transfer */

int main(int argc, char **argv) {
int c, s, bytes;
char buf[BUF_SIZE]; struct     /* zona tampon de memorie pentru fişierul ce este recepţionat */
hostent *h;
                                                       /* informaţii despre server */
  struct sockaddr_în channel;
                                                              /* păstrează adresa IP */
if (argc != 3) fatal("Usage: client server-name file-name"); h = gethostbyname(argv[1]);
if (!h) fatal("gethostbyname failed");
                                                       /* caută adresa IP a gazdei */

s = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
if (s < 0) fatal("socket");
memset(&channel, 0, sizeof(channel));
channel.sin_family= AF_INET;
memcpy(&channel.sin_addr.s_addr, h->h_addr, h->h_length); channel.sin_port=
htons(SERVER_PORT);

c = connect(s, (struct sockaddr *) &channel, sizeof(channel));
if (c < 0) fatal("connect failed");

/* Conexiunea este acum stabilită. Trimite numele fişierului incluzând terminatorul de şir */
write(s, argv[2], strlen(argv[2])+1);

/* Ia fişierul şi-l afişează la ieşirea standard. */
while (1) {
    bytes = read(s, buf, BUF_SIZE);                                  /* citeşte de la soclu */
    if (bytes <= 0) exit(0);                                /* verifică dacă este sfârşit de fişier*/
    write(1, buf, bytes);
                                                          /* scrie la ieşirea standard */


fatal(char *string)
{
printf("%s\n", string); exit( 1 );
}
```

Fig. 6-6. Codul client folosind socluri. Codul server este pe pagina următoare.

```
#include <sys/types.h>
#include <sys/fcntl.h>
#include <sys/socket.h> #include <netinet/în.h>
#include <netdb.h>
#define SERVER_PORT 12345       /* arbitrar, dar clientul şi serverul trebuie să fie de acord */
#define BUF_SIZE 4096                                   /* dimensiunea blocului de transfer */
#define QUEUE_SIZE 10

int main(int argc, char *argv[])
{
int s, b, l, fd, sa, bytes, on = 1;
char buf[BUF_SIZE]; /* zona tampon de memorie pentru fişierul care este transmis */ struct
sockaddr_în channel;
                                                                /* păstrează adresa IP */

/* Construieşte structura adresei pentru a se lega la soclu. */
memset(&channel, 0, sizeof(channel));                                   /* canalul zero */
channel.sin_family = AF_INET;
channel.sin_addr.s_addr = htonl(INADDR_ANY); channel.sin_port =
htons(SERVER_PORT);

/* Deschidere pasivă. Aşteaptă conexiunea. */
s = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);                          /* creează soclu */
if (s < 0) fatal("socket failed");
setsockopt(s, SOL_SOCKET, SO_REUSEADDR, (char *) &on, sizeof(on));

b = bind(s, (struct sockaddr *) &channel, sizeof(channel));
if (b < 0) fatal("bind failed");

l = listen(s, QUEUE_SIZE); if                           /* specifică dimensiunea cozii */
(l < 0) fatal("listen failed");

/* Soclul este acum setat şi legat. Aşteaptă conexiunea şi o procesează.
*/ while (1) {
   sa = accept(s, 0, 0);                          /* blocare pentru cererea de conexiune */
   if (sa < 0) fatal("accept failed");

   read(sa, buf, BUF_SIZE);                        /* citeşte numele fişierului de la soclu */

   /* Preia şi returnează fişierul. */ fd = open(buf, O_RDONLY); if (fd < 0) fatal("open failed");
                                                    /* deschide fişierul de trimis înapoi */
   while (1) {
      bytes = read(fd, buf, BUF_SIZE); /* citeşte din fişier */
      if (bytes <= 0) break;                        /* verifică dacă este sfârşit de fişier */
      write(sa, buf, bytes);                                /* scrie octeti la soclu */
   } close(fd); close(sa);
}
}
                                                                * închide fişierul */ /*
                                                               închide conexiunea*/
```

Fig. 6-6. Codul server. Codul client este pe pagina

În continuare serverul creează un soclu şi face verificare pentru erori (indicate de *s*<0). Într-o versiune de producţie a codului mesajul de eroare ar putea fi mai

explicit. Apelul către *setsockopt* este necesar pentru a permite portului să fie folosit de serverul care rulează la nesfârşit, răspunzând la cerere după cerere. Acum, adresa IP este legată la soclu şi este făcută o verificare pentru a vedea dacă apelul către *bind* a reuşit. Ultimul pas în iniţializare este apelul către *listen* pentru a anunţa acordul serverului de a accepta apeluri şi pentru a spune sistemului să menţină un număr de până la *QUEUE_SIZE* din acestea în caz că ajung noi cereri în timp ce serverul o procesează încă pe cea curentă. Dacă coada este plină şi ajung cereri suplimentare, se renunţă la acestea.

În acest punct, serverul intră în bucla sa principală, pe care nu o mai părăseşte niciodată. Singura cale de a-l opri este de a-l termina forţat din afară. Apelul la *accept* blochează serverul până când un client încearcă să stabilească o conexiune cu el. Dacă apelul *accept* e făcut cu succes, returnează un descriptor de fişier care poate fi folosit pentru citire şi scriere, în mod asemănător descriptorilor ce sunt folosiţi pentru a citi şi a scrie în pipe-uri (tuburi). Cu toate acestea, spre deosebire de tuburi, care sunt unidirecţionale, soclurile sunt bidirecţionale, astfel că *sa* (socket address – adresa soclului) poate fi folosită şi pentru citire din conexiune, şi pentru a scrie pe ea.

După ce conexiunea este stabilită, serverul citeşte numele fişierului din ea. Dacă numele nu este încă disponibil, serverul se blochează aşteptându-l. După ce ia numele fişierului, serverul deschide fişierul şi intră într-o buclă care citeşte alternativ blocuri din fişier şi le scrie pe soclu până când întregul fişier a fost copiat. Apoi serverul închide iar fişierul şi conexiunea şi aşteaptă să apară următoarea conexiune. El repetă această buclă la infinit.

Acum să privim codul client. Pentru a înţelege cum funcţionează, este necesar să înţelegem cum este invocat. Presupunând că este numit *client*, un apel tipic este

```
client flits.cs.vu.nl /usr/tom/filename >f
```

Acest apel funcţionează doar dacă serverul rulează deja la *flits.cs.vu.nl* şi fişierul *lusr/tomlfilemane* există şi serverul are drept de citire pentru el. Dacă apelul are succes, fişierul este transferat prin Internet şi scris în *f* , după care programul client se termină. Din moment ce serverul continuă după un transfer, clientul poate fi pornit din nou pentru a lua alte fişiere.

Codul client începe cu câteva directive *include* şi declaraţii. Execuţia începe verificând dacă a fost apelat cu număr corect de argumente *(argc*=3 înseamnă numele programului plus două argumente). Observaţi că *argv[1]conţine* numele serverului (de exemplu *flits.cs.vu.nl)* şi este convertit la o adresă IP către *gethostbyname.* Această funcţie foloseşte DNS pentru a căuta numele. Vom studia DNS în cap. 7.

În continuare este creat şi iniţializat un soclu. Apoi, clientul încearcă să stabilească o conexiune TCP cu serverul, folosind *connect.* Dacă serverul funcţionează pe maşina menţionată şi ataşat la *SERVER_PORT* şi este fie inactiv, fie are loc în coada sa *listen,* conexiunea va fi (în cele din urmă) stabilită. Folosind conexiunea, clientul trimite numele fişierului scriind pe soclu. Numărul de octeţi trimişi este cu 1 mai mare decât numele, deoarece terminatorul de şir (un octet 0) trebuie de asemenea trimis pentru a spune serverului unde se sfârşeşte numele.

Acum clientul intră într-o buclă, citind fişierul bloc cu bloc de la soclu şi copiindu-l la ieşirea standard. Când acestea se termină, pur şi simplu iese.

Procedura *fatal* afişează un mesaj de eroare şi iese. Serverul are nevoie de aceeaşi procedură, dar aceasta a fost omisă datorită lipsei de spaţiu pe pagina. Din moment ce clientul şi serverul sunt compilate separat şi în mod normal rulează pe calculatoare diferite, ele nu pot partaja codul procedurii *fatal*.

Aceste două programe (la fel ca şi orice material referitor la această carte) pot fi luate de la adresa de Web a cărţii

http://www.prenhall.com/tanenbaum

dând clic pe link-ul către situl Web de lângă fotografia coperţii. Ele pot fi descărcate şi compilate pe orice sisteme UNIX (de exemplu Solaris, BSD, Linux) cu comenzile:

```
cc -o client client.c -lsocket -lnsl
cc -o server sever.c -lsocket -lnsl
```

Serverul este pornit tastând doar

```
server
```

Clientul are nevoie de două argumente, aşa cum s-a discutat mai sus. O versiune de Windows este de asemenea disponibilă pe situl Web.

Ca o observaţie, acest server nu este ultimul cuvânt în domeniul programelor server. Verificarea erorilor este ineficientă şi raportarea erorilor este mediocră. În mod clar serverul nu a auzit niciodată de securitate, şi folosirea doar a apelurilor de sistem UNIX nu este ultimul cuvânt în independenţa de platformă. De asemenea face unele presupuneri care sunt tehnic ilegale, cum ar fi presupunerea că numele fişierului încape în zona de memorie tampon şi este transmis automat. Din moment ce tratează toate cererile strict secvenţial (deoarece are doar un singur fir de execuţie) performanţa este slabă. În ciuda acestor neajunsuri, este un server de fişiere Internet complet şi funcţional. În exerciţii, cititorul este invitat să le îmbunătăţească. Pentru mai multe informaţii despre programare cu socluri, a se vedea (Stevens, 1997).

6.2 NOŢIUNI DE BAZĂ DESPRE PROTOCOALELE DE TRANSPORT

Serviciul transport este implementat prin intermediul unui **protocol de transport** folosit de cele două entităţi de transport. Câteva caracteristici sunt asemănătoare pentru protocoalele de transport şi pentru cele de legătură de date studiate în detaliu în cap. 3. Amândouă trebuie să se ocupe, printre altele, de controlul erorilor, de secvenţiere şi de controlul fluxului.

Totuşi, există diferenţe semnificative între cele două protocoale. Aceste

diferenţe sunt datorate deosebirilor majore dintre mediile în care operează protocoalele, aşa cum rezultă din fig. 6-7. La nivelul legăturii de date, cele două rutere comunică direct printr-un canal fizic, în timp ce la nivelul transport acest canal fizic este înlocuit de întreaga subreţea. Această deosebire are mai multe implicaţii importante pentru protocoale, aşa cum vom vedea în acest capitol.

În cazul legăturii de date, pentru un ruter nu trebuie specificat cu care alt ruter vrea să comunice, deoarece fiecare linie specifică în mod unic o destinaţie. În schimb, în cazul nivelului transport este necesară adresarea explicită.

În plus, procesul stabilirii unei conexiuni prin cablul din fig. 6-7(a) este simplu: celălalt capăt este întotdeauna acolo (în afară de cazul în care nu a ‘căzut’) şi în nici unul din cazuri nu sunt prea multe de făcut. Pentru nivelul transport însă, stabilirea iniţială a conexiunii este mult mai complicată, aşa cum vom vedea.

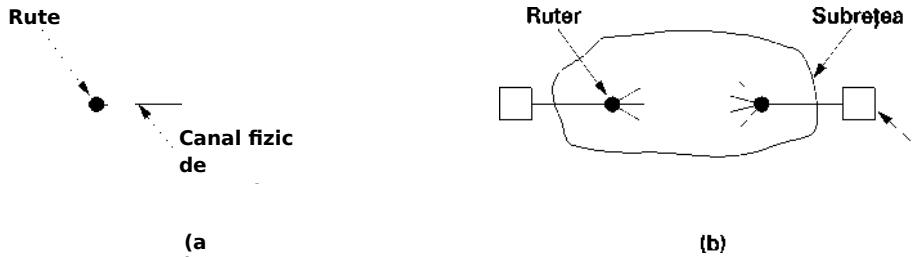


Fig. 6-7. (a) Mediul pentru nivelul legătură de date. (b) Mediul pentru nivelul transport.

O altă diferenţă între nivelurile legătură de date şi transport, care generează multe probleme, este existenţa potenţială a unei capacităţi de memorare a subreţelei. Atunci când un ruter trimite un cadru (nivel legătură de date), acesta poate să ajungă sau poate să se piardă, dar nu poate să se plimbe un timp ajungând până la capătul lumii şi să apară 30 de secunde mai târziu, într-un moment nepotrivit. Dacă subreţeaua foloseşte datagrame şi dirijare adaptivă, există o posibilitate – care nu poate fi neglijată – ca un pachet să fie păstrat pentru un număr oarecare de secunde şi livrat mai târziu. Consecinţele capacităţii de memorare a subreţelei pot fi uneori dezastruoase şi necesită folosirea unor protocoale speciale.

O ultimă diferenţă între nivelurile legătură de date şi transport este una de dimensionare şi nu de proiectare. Folosirea tampoanelor şi controlul fluxului sunt necesare la amândouă nivelurile, dar prezenţa unui număr mare de conexiuni în cazul nivelului transport necesită o abordare diferită de cea de la nivelul legătură de date. În cap. 3, unele protocoale alocau un număr fix de tampoane pentru fiecare linie, astfel încât atunci când sosea un cadru, exista întotdeauna un tampon disponibil. La nivel transport, numărul mare de conexiuni care trebuie să fie gestionate face ca ideea de a aloca tampoane dedicate să fie mai puţin atractivă. În următoarele secţiuni, vom examina atât aceste probleme importante cât şi altele.

6.2.1 Adresarea

Atunci când un proces aplicaţie (de exemplu, un proces utilizator) doreşte să stabilească o conexiune cu un proces aflat la distanţă, el trebuie să specifice cu care proces doreşte să se conecteze. (La protocoalele de transport neorientate pe conexiune apare aceeaşi problemă: cui trebuie trimis mesajul?). Metoda folosită în mod normal este de a defini adrese de transport la care procesele pot să aştepte cereri de conexiune. În Internet acestea se numesc porturi. La reţelele ATM perechile se numesc **AAL – SAP**-uri. În continuare vom folosi pentru acestea termenul generic **TSAP (Transport Service Access Point –** punct de acces la serviciul de transport). Punctele similare în cazul nivelului reţea (adică adresele la nivel reţea) sunt numite **NSAP (Network Service Access Point).** Adresele IP sunt exemple de NSAP-uri.

Fig. 6-8 ilustrează relaţia între TSAP, NSAP şi conexiunile transport. Procesele aplicaţie, atât cli- enţii cât şi serverele, se pot ataşa la TSAP pentru a stabili o

conexiune la un TSAP la distanţă. Aceste conexiuni rulează prin TSAP-uri pe fiecare gazdă aşa cum se arată. Necesitatea de a avea mai multe TSAP-uri este dată de faptul că în unele reţele, fiecare calculator are un singur NSAP, deci cumva este nevoie să se distingă mai multe puncte de sfârşit de transport care partajează acel NSAP.

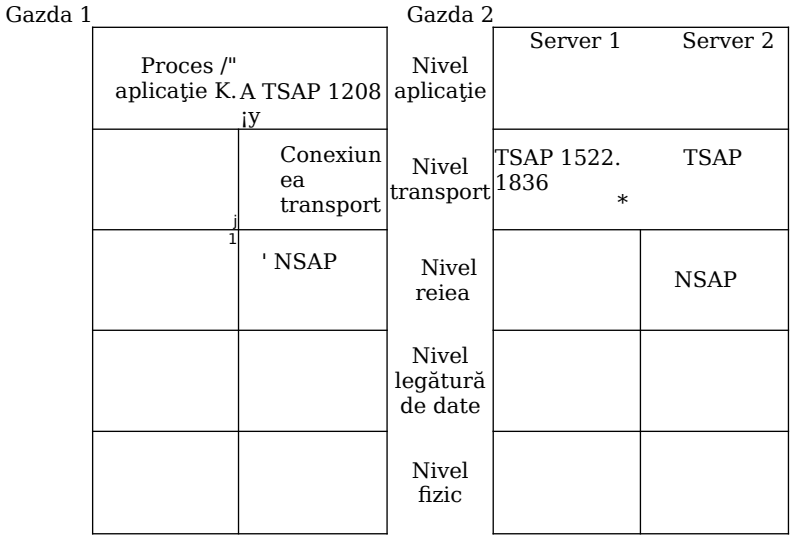


Fig. 6-8. TSAP, NSAP şi conexiunile la nivel transport.

Un scenariu posibil pentru stabilirea unei conexiuni la nivel transport este următorul.

1. Un proces server care furnizează ora exactă şi care rulează pe gazda 2 se ataşează la TSAP 122 aşteptând un apel. Felul în care un proces se ataşează la un TSAP nu face parte din modelul de reţea şi depinde numai de sistemul de operare local. Poate fi utilizat un apel de tip LISTEN din capitolul precedent.
2. Un proces aplicaţie de pe gazda 1 doreşte să afle ora exactă; atunci el generează un apel CONNECT specificând TSAP 1208 ca sursă şi TSAP 1522 ca destinaţie. Această acţiune are ca rezultat în cele din urmă stabilirea unei conexiuni la nivel transport între procesele aplicaţie de pe gazda 1 şi serverul 1 de pe gazda 2.
3. Procesul aplicaţie trimite o cerere o cerere pentru timp.
4. Procesul server de timp răspunde cu timpul curent.
5. Conexiunea transport este apoi eliberată.

Trebuie reţinut că foarte bine pot exista alte servere pe gazda 2 care să fie ataşate la alte TSAP- uri şi care să aştepte conexiuni care ajung pe acelaşi NSAP.

Fig. 6-8 explică aproape tot, cu excepţia unei mici probleme: cum ştie procesul utilizator de pe maşina 1 că serverul de oră exactă este ataşat la TSAP 1522? O

posibilitate este ca acest server de oră exactă să se atașeze la TSAP 1522 de ani de zile și, cu timpul, toți utilizatorii au aflat acest lucru. În acest model serviciile au adrese TSAP fixe, care pot fi afișate în fișiere în locuri bine cunoscute, cum este fișierul *etc/services* pe sistemele UNIX care afișează ce servere sunt atașate permanent și la ce porturi.

Dar schema cu adrese de servicii fixe funcționează doar pentru un număr mic de servicii cheie, a căror adresă nu se schimbă niciodată (de exemplu server de Web). Însă, în general, procesele utilizator vor să comunice cu alte procese care există numai pentru scurt timp și nu au o adresă TSAP dinainte cunoscută. Pe de altă parte, pot exista mai multe procese server, majoritatea utilizate foarte

rar, şi ar fi neeconomic ca fiecare să fie activ şi să asculte la o adresă TSAP fixă tot timpul. Pe scurt, este necesară o soluţie mai bună.

O astfel de soluţie este prezentată în fig. 6-9, într-o formă simplificată. Ea este cunoscută ca **protocolul de conectare iniţială.** În loc ca orice server să asculte la un TSAP fixat, fiecare maşină care doreşte să ofere servicii utilizatorilor aflaţi la distanţă are un **server de procese (process server)** special care acţionează ca un intermediar pentru toate serverele mai puţin utilizate. El ascultă în acelaşi timp la un număr de porturi, aşteptând o cerere de conexiune. Utilizatorii potenţiali ai serviciului încep prin a face o cerere de conexiune, specificând adresa TSAP a serviciului pe care îl doresc. Dacă nu există un server care să aştepte conexiuni la acel port, ele obţin o conexiune la serverul de procese, ca în fig. 6-9 (a).

După ce primeşte cererea, serverul de procese dă naştere serverului cerut, permiţându-i să moştenească conexiunea cu procesul utilizator. Noul server execută prelucrarea cerută, în timp ce serverul de procese continuă să aştepte noi cereri, ca în fig. 6-9 (b).

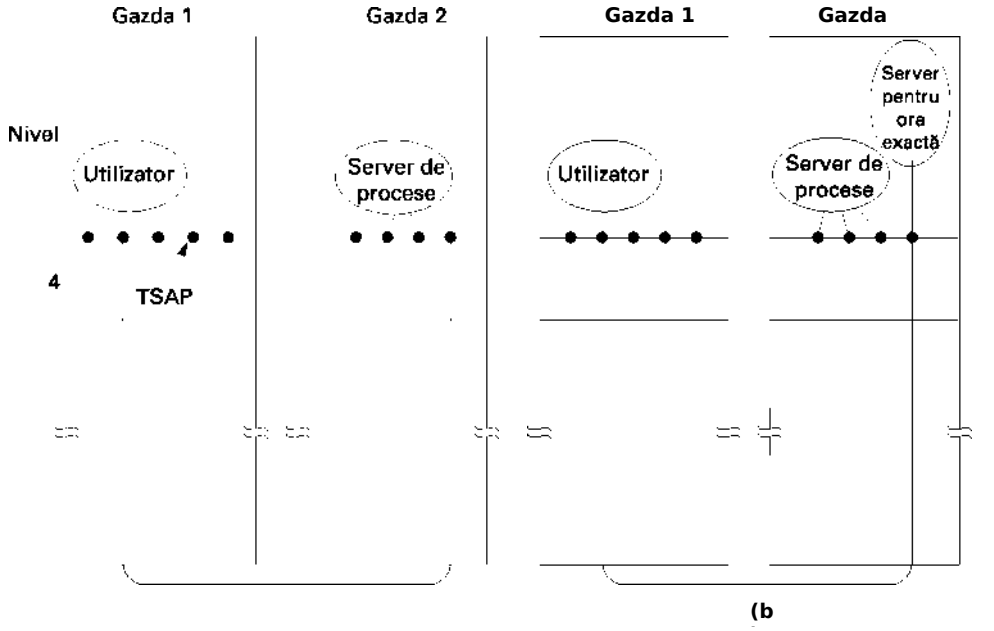


Fig. 6-9. Stabilirea unei conexiuni între calculatorul gazdă 1 şi serverul pentru ora

În timp ce acest protocol funcţionează bine pentru serverele care pot fi create ori de câte ori este nevoie de ele, există mai multe situaţii în care serviciile există independent de serverul de procese. De exemplu, un server de fişiere va rula folosind un hardware specializat (o maşina cu disc) şi nu poate fi creat din mers.

Pentru a trata această situaţie, este des utilizată o soluţie alternativă. În acest model există un proces special numit **server de nume** (**name server** sau, uneori, **directory server**). Pentru a găsi adresa TSAP corespunzătoare unui serviciu dat

prin nume, aşa cum este „ora exactă”, utilizatorul stabileşte o conexiune cu serverul de nume (care aşteaptă mesaje la un TSAP cunoscut). Apoi utilizatorul trimite un mesaj specificând numele serviciului, iar serverul de nume îi trimite înapoi adresa TSAP a acestuia. După aceasta, utilizatorul eliberează conexiunea cu serverul de nume şi stabileşte o nouă conexiune cu serviciul dorit.

În acest model, atunci când este creat un nou serviciu, el trebuie să se înregistreze singur la serverul de nume, furnizând atât numele serviciului oferit (în general un şir ASCII) cât şi adresa TSAP. Serverul de nume înregistrează această informaţie într-o bază de date internă, astfel încât el va şti răspunsul atunci când vor sosi noi cereri.

Funcţionarea serverului de nume este asemănătoare cu serviciul de informaţii de la un sistem telefonic: este furnizată corespondenţa dintre nume şi numere de telefon. Ca şi în cazul telefoanelor, este esenţial ca adresa bine cunoscută a serverului de nume (sau a serverului de procese, în protocolul de conectare iniţială) să fie într-adevăr bine cunoscută. Dacă nu ştii numărul de la informaţii, nu poţi afla nici un alt număr de telefon. Dacă crezi că numărul de la informaţii este evident pentru toţi, încearcă să-l foloseşti şi în altă ţară!

6.2.2 Stabilirea conexiunii

Stabilirea unei conexiuni poate să pară uşoară dar, în realitate, este surprinzător de complicată. La prima vedere, ar părea suficient ca o entitate de transport să trimită numai un TPDU CONNECTION REQUEST şi să aştepte replica CONNECTION ACCEPTED . Problema apare deoarece reţeaua poate pierde, memora sau duplica pachete. Acest comportament duce la complicaţii serioase.

Putem imagina o subreţea care este atât de congestionată încât confirmările ajung greu înapoi, şi, din această cauză, fiecare pachet ajunge să fie retransmis de câteva ori. Putem presupune că subre- ţeaua foloseşte datagrame şi fiecare pachet urmează un traseu diferit. Unele pachete pot să întâlnească o congestie locală de trafic şi să întârzie foarte mult, ca şi cum ar fi fost memorate de subreţea un timp şi eliberate mai târziu.

Cel mai neplăcut scenariu ar fi: un utilizator stabileşte o conexiune cu o bancă şi trimite un mesaj cerând transferul unei sume de bani în contul unei alte persoane în care nu poate avea încredere în totalitate, şi apoi eliberează conexiunea. Din nefericire, fiecare pachet din acest scenariu este duplicat şi memorat în subreţea. După ce conexiunea a fost eliberată, pachetele memorate ies din subre- ţea şi ajung la destinatar, cerând băncii să stabilească o nouă conexiune, să facă transferul (încă o dată) şi să elibereze conexiunea. Banca nu poate să ştie că acestea sunt duplicate, ea trebuie să presupună că este o tranzacţie independentă şi va transfera banii încă o dată. În continuarea acestei secţiuni, vom studia problema duplicatelor întârziate, punând accentul în mod special pe algoritmii pentru stabilirea sigură a conexiunilor, astfel încât scenarii ca cel de mai sus să nu poată să apară.

După cum am mai spus, punctul crucial al problemei este existenţa duplicatelor întârziate. El poate fi tratat în mai multe feluri, dar nici unul nu este într-adevăr satisfăcător. O posibilitate este de a utiliza adrese de transport valabile doar pentru

o singură utilizare. În această abordare, ori de câte ori este necesară o adresă la nivel transport, va fi generată una nouă. După ce conexiunea este eliberată, adresa nu mai este folosită. Acest mecanism face însă imposibl modelul cu server de procese din fig. 6-9.

O altă posibilitate este de a atribui fiecărei conexiuni un identificator (adică, un număr de secvenţă incrementat pentru fiecare conexiune stabilită), ales de cel care iniţiază conexiunea, şi pus în fiecare TPDU, inclusiv în cel care iniţiază conexiunea. După ce o conexiune este eliberată, fiecare entitate de transport va completa o tabelă cu conexiunile care nu mai sunt valide, reprezentate ca perechi (entitate de transport, identificator conexiune). Ori de câte ori apare o cerere de conexiune se va verifica în tabelă că ea nu aparţine unei conexiuni care a fost eliberată anterior.

Din nefericire, această schemă are un defect important: ea necesită ca fiecare entitate de transport să menţină informaţia despre conexiunile precedente un timp nedefinit. Dacă o maşină cade şi îşi pierde datele din memorie, ea nu va mai şti care identificatori de conexiune au fost deja utilizaţi.

Putem încerca şi o altă soluţie. În loc să permitem pachetelor să trăiască la nesfârşit în subreţea, putem inventa un mecanism care să elimine pachetele îmbătrânite. Dacă suntem siguri că nici un pachet nu poate să supravieţuiască mai mult de un anume interval de timp cunoscut, problema devine ceva mai uşor de rezolvat.

Durata de viaţă a pachetelor poate fi limitată la un maxim cunoscut, folosind una (sau mai multe) din următoarele tehnici:

1. Restricţii în proiectarea subreţelei
2. Adăugarea unui contor al nodurilor parcurse în fiecare pachet
3. Adăugarea unei amprente de timp la fiecare pachet

Prima metodă include soluţiile care împiedică pachetele să stea în buclă, combinate cu modalităţi de a limita întârzierile datorate congestiilor, pe orice cale din reţea (indiferent de lungime). A doua metodă constă în a iniţializa contorul cu o valoare adecvată şi în a-l decrementa la trecerea prin orice nod. Protocolul de nivel reţea pur şi simplu elimină pachetele al căror contor a devenit zero. A treia metodă presupune ca fiecare pachet să conţină timpul creării sale, ruterele acceptând să elimine pachetele mai vechi de un anumit moment de timp, asupra căruia au căzut de acord. Această metodă necesită ca ceasurile de la fiecare ruter să fie sincronizate, şi această cerinţă în sine este destul de greu de îndeplinit (mai uşor este dacă sincronizarea ceasurilor se obţine din exteriorul reţelei, de exemplu folosind GPS sau staţii radio care transmit periodic ora exactă).

În practică, nu este suficient doar să garantăm că pachetul este eliminat, ci trebuie garantat şi că toate confirmările sale au fost eliminate, astfel încât vom introduce *T,* care va fi un multiplu (mic) al duratei maxime de viaţă a unui pachet. Depinde de protocol de câte ori *T* este mai mare decât durata de viaţă a unui pachet. Dacă aşteptăm un timp *T* după trimiterea unui pachet putem fi siguri că toate urmele sale au dispărut şi nici el, nici vreo confirmare de-a sa nu vor apărea din senin, doar ca să complice lucrurile.

Folosind durata de viaţă limitată a pachetelor, există metode de a obţine conexiuni sigure a căror corectitudine a fost demonstrată. Metoda descrisă în cele ce urmează este datorată lui Tomlinson (1975). Ea rezolvă problema, dar introduce câteva particularităţi proprii. Metoda a fost îmbunătăţită de Sunshine şi Dalal (1978). Variante ale sale sunt larg folosite în practică, inclusiv în TCP.

Pentru a ocoli problemele generate de pierderea tuturor datelor din memoria unei maşini după o cădere, Tomlinson propune echiparea fiecărei maşini cu un ceas. Nu este nevoie ca ceasurile de pe maşini diferite să fie sincronizate. Fiecare ceas va fi de fapt un contor binar care se autoincrementea- ză după un anumit interval de timp. În plus, numărul de biţi ai contorului trebuie să fie cel puţin egal cu numărul de biţi al numerelor de secvenţă. În cele din urmă, şi cel mai important, ceasul trebuie să continue să funcţioneze chiar în cazul în care calculatorul gazdă cade.

Ideea de bază este de a fi siguri că două TPDU numerotate identic nu pot fi generate în acelaşi timp. Atunci când conexiunea este iniţiată, k biţi mai puţin semnificativi ai ceasului sunt folosiţi ca număr iniţial de secvenţă (tot k biţi). Astfel, fiecare conexiune începe să-şi numeroteze TPDU-urile sale cu un număr de secvenţă diferit. Spaţiul numerelor de secvenţă ar trebui să fie suficient de mare pentru ca, în timpul scurs până când contorul ajunge din nou la acelaşi număr, toate TPDU-urile vechi cu acel număr să fi dispărut deja. Această relaţie liniară între timp şi numărul de secvenţă iniţial este prezentată în fig. 6-10.

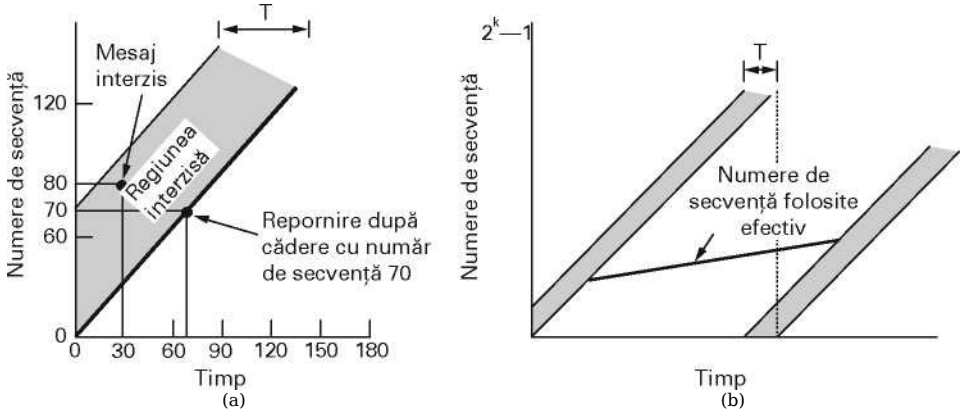


Fig. 6-10. (a) TPDU-urile nu pot să intre în regiunea interzisă. (b) Problema resincronizării.

Odată ce ambele entităţi de transport au căzut de acord asupra numărului de secvenţă iniţial, pentru controlul fluxului poate fi folosit orice protocol cu fereastră glisantă. În realitate curba ce reprezintă numărul iniţial de secvenţă (desenată cu linie îngroşată) nu este chiar liniară, ci în trepte, căci ceasul avansează în trepte. Pentru simplitate, vom ignora acest detaliu.

O problemă apare atunci când cade un calculator gazdă. Când el îşi revine, entitatea sa de transport nu ştie unde a rămas în spaţiul numerelor de secvenţă. O soluţie este de a cere entităţii de transport să stea neocupată T secunde după revenire pentru ca în acest timp toate vechile TPDU să dispară. Totuşi, într-o reţea

complexă *T* poate fi destul de mare, astfel că această strategie nu este prea atrăgătoare.

Pentru a evita cele *T* secunde de timp nefolosit după o cădere, este necesar să introducem o nouă restricţie în utilizarea numerelor de secvenţă. Necesitatea introducerii acestei restricţii este evidentă în următorul exemplu. Fie *T*, timpul maxim de viaţă al unui pachet, egal cu 60 de secunde şi să presupunem că ceasul este incrementat la fiecare secundă. După cum arată linia îngroşată din fig. 6-10(a), numărul iniţial de secvenţă pentru o conexiune iniţiată la momentul *x* este *x.* Să ne imaginăm că la t=30 sec, unui TPDU trimis pe conexiunea cu numărul 5 (deschisă anterior) i se dă numărul de secvenţă 80. Să numim acest TPDU *X.* Imediat după ce *X* este trimis, calculatorul gazdă cade şi revine imediat. La t=60 el redeschide conexiunile de la 0 la 4. La t=70, el deschide conexiunea 5, folosind un număr de secvenţă iniţial 70, aşa cum am stabilit. În următoarele 15 secunde el va transmite TPDU-uri cu date numerotate de la 70 la 80. Astfel că la t=85, în subreţea este generat un nou TPDU cu numărul de secvenţă 80 şi conexiunea 5. Din nefericire, TPDU *X* încă mai există. Dacă el ajunge înaintea noului TPDU 80, atunci TPDU *X* va fi acceptat şi TPDU-ul corect va fi respins ca fiind un duplicat.

Pentru a preveni o astfel de problemă trebuie să luăm măsuri ca numerele de secvenţă să nu fie utilizate (adică atribuite unor noi TPDU-uri) un timp T înaintea utilizării lor ca noi numere de secvenţă. Combinaţiile imposibile – timp, număr de secvenţă – sunt prezentate în fig. 6-10(a) ca **regiunea interzisă**. Înainte de trimiterea oricărui TPDU pe orice conexiune, entitatea de transport trebuie să citească ceasul şi să verifice dacă nu cumva se află în regiunea interzisă.

Pot să apară probleme în două cazuri: dacă un calculator gazdă trimite prea multe date şi prea repede pe o conexiune nou deschisă, curba numărului de secvenţă în funcţie de timp poate să fie mult mai abruptă decât cea iniţială. Aceasta înseamnă că rata de transmisie pentru orice conexiune este de cel mult un TPDU pe unitatea de timp a ceasului. De asemenea, este necesar ca entitatea de transport să aştepte până când ceasul avansează o dată, înainte să deschidă o nouă conexiune pentru ca, la revenirea după o cădere, acelaşi număr de secvenţă să nu fie utilizat de două ori. Cele două observaţii de mai sus sunt argumente pentru ca perioada ceasului să fie cât mai mică (câteva LLS sau mai mică).

Din nefericire, intrarea în regiunea interzisă prin trimitere prea rapidă nu este singura situaţie care creează probleme. Fig. 6-10(b) arată că la orice rată de transfer mai mică decât frecvenţa ceasului curba numerelor de secvenţă utilizate raportată la timp va ajunge până la urmă în regiunea interzisă din stânga. Cu cât curba numerelor de secvenţă utilizate va fi mai înclinată, cu atât mai târziu se ajunge în regiunea interzisă. Aşa cum am afirmat anterior, imediat înaintea trimiterii unui TPDU, entitatea de transport trebuie să verifice dacă nu se află cumva în regiunea interzisă, şi, dacă se află, să întârzie transmisia cu T secunde sau să resincronizeze numerele de secvenţă.

Metoda bazată pe ceasuri rezolvă problema duplicatelor întârziate pentru TPDU-urile de date, dar pentru ca această metodă să poată fi folosită, trebuie mai întâi să stabilim conexiunea. Deoarece TPDU-urile de control pot şi ele să fie întârziate, pot apărea probleme atunci când entităţile de transport încercă să cadă de acord

asupra numărului iniţial de secvenţă. Să presupunem, de exemplu, că, pentru a stabili o conexiune, gazda 1 trimite un mesaj CONNECTION REQUEST conţinând numărul de secvenţă iniţial propus şi portul destinaţie gazdei 2. Acesta va confirma mesajul trimiţând înapoi un TPDU CONNECTION ACCEPTED . Dacă TPDU-ul CONNECTION REQUEST este pierdut, dar un duplicat întârziat al unui alt CONNECTION REQUEST va ajunge la gazda 2, atunci conexiunea nu va fi stabilită corect.

Pentru a rezolva aceasta problemă, Tomlinson (1975) a introdus stabilirea conexiunii cu înţelegere în trei paşi (three-way handshake). Acest protocol nu necesită ca ambele părţi să înceapă să trimită acelaşi număr de secvenţă, deci poate fi utilizat şi împreună cu alte metode de sincronizare decât ceasul global. Procedura normală de iniţiere a conexiunii este exemplificată în fig. 6-11(a). Gazda 1 alege un număr de secvenţă *x* şi trimite un TPDU CONNECTION REQUEST care conţine *x* gazdei 2. Gazda 2 răspunde cu CONNECTION ACK, confirmând *x* şi anunţând numărul său iniţial de secvenţă, *y.* În cele din urmă gazda 1 confirmă alegerea lui *y* gazdei 2 în primul mesaj de date pe care îl trimite.

Vom arunca acum o privire asupra stabilirii conexiunii cu înţelegere în trei paşi în prezenţa TPDU-urilor de control duplicate întârziate. În fig. 6-11(b) primul TPDU sosit este o copie întârziată a unui CONNECTION REQUEST de la o conexiune mai veche. Acest TDU ajunge la gazda 2 fără ca gazda 1 să ştie. Gazda 2 răspunde acestui TPDU trimiţând gazdei 1 un TPDU ACK, verificând de fapt că gazda 1 a încercat într-adevăr să stabilească o conexiune. Atunci când gazda 1 refuză cererea gazdei 2 de a stabili conexiunea, gazda 2 îşi dă seama că a fost păcălită de o copie întârziată şi abandonează conexiunea. În acest fel o copie întârziată nu poate să strice nimic.

În cel mai rău caz, atât CONNECTION REQUEST cât şi ACK sunt copii întârziate în subreţea. Acest caz este prezentat în 6-11(c). Ca şi în exemplul precedent, gazda 2 primeşte o comandă CONNECTION REQUEST întârziată şi răspunde la ea. În acest moment este extrem de important să ne aducem aminte că gazda 2 a propus *y* ca număr iniţial de secvenţă pentru traficul de la 2 la 1, fiind sigur că nu mai există în reţea nici un TPDU (sau confirmare) cu acelaşi număr de secvenţă. Atunci când al doilea TPDU întârziat ajunge la gazda 2, aceasta deduce, din faptul că a fost confirmat *z* şi nu *y,* că are de-a face cu o copie mai veche. Important este că nu există nici o combinaţie posibilă ale unor copii vechi ale TPDU-urilor întârziate care să reuşească să iniţieze o conexiune atunci când nimeni nu a cerut asta.

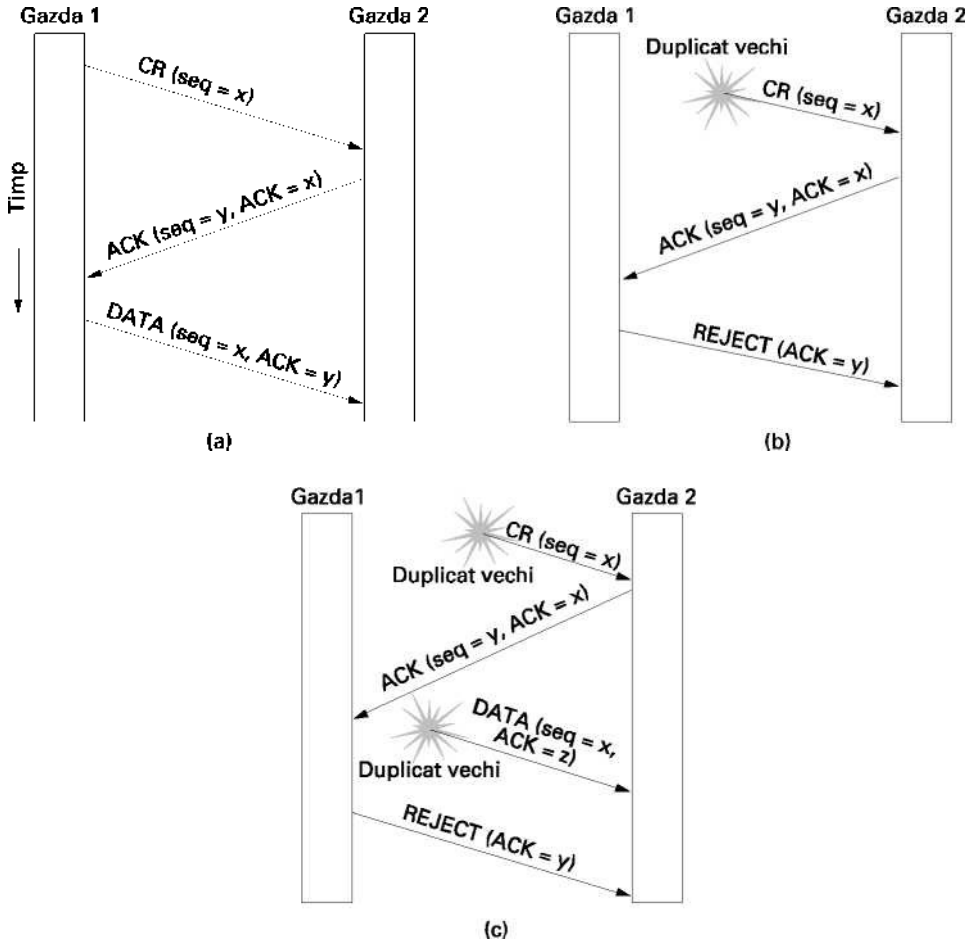


Fig. 6-11. Trei scenarii posibile de stabilire a conexiunii pentru un protocol cu înţelegere în trei paşi. CR reprezintă CONNECTION REQUEST. (a) Cazul normal, (b) Un duplicat vechi al unui mesaj CONNECTION REQUEST apare când nu trebuie, (c) Sunt duplicate atât CONNECTION REQUEST cât şi CONNECTION

6.2.3 Eliberarea conexiunii

Eliberarea unei conexiuni este mai uşoară decât stabilirea ei. Totuşi, există mai multe dificultăţi decât ne-am aştepta. Aşa cum am mai amintit, există două moduri de a termina o conexiune: eliberare simetrică şi eliberare asimetrică. Sistemul telefonic foloseşte eliberarea asimetrică: atunci când unul din interlocutori închide, conexiunea este întreruptă. Eliberarea simetrică priveşte conexiunea ca pe două conexiuni separate unidirecţionale şi cere ca fiecare să fie eliberată separat.

Eliberarea asimetrică este bruscă şi poate genera pierderi de date. Să considerăm scenariul din fig. 6-12. După stabilirea conexiunii, gazda 1 trimite un TPDU care ajunge corect la gazda 2. Gazda 1 mai trimite un TPDU dar, înainte ca acesta să ajungă la destinaţie, gazda 2 trimite DISCONNECT REQUEST . În acest

caz, conexiunea va fi eliberată şi vor fi pierdute date.

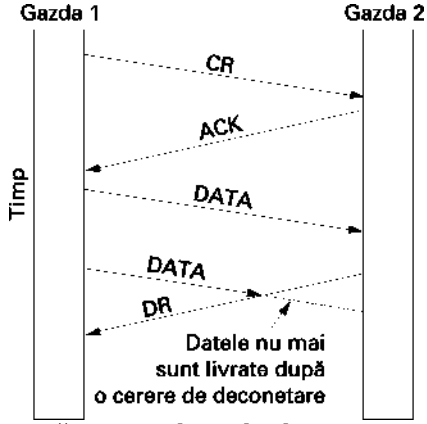


Fig. 6-12. Deconectare bruscă cu pierdere de date. CR= CONNECTION REQUEST, ACK=CONNECTION ACCEPTED , DR=DISCONNECT REQUEST.

Evident, pentru a evita pierderea de date, este necesar un protocol de eliberare a conexiunii mai sofisticat. O posibilitate este utilizarea eliberării simetrice: fiecare direcţie este eliberată independent de cealaltă; un calculator gazdă poate să continue să primească date chiar şi după ce a trimis un TPDU de eliberare a conexiunii.

Eliberarea simetrică este utilă atunci când fiecare proces are o cantitate fixă de date de trimis şi ştie bine când trebuie să transmită şi când a terminat. În alte situaţii însă, nu este deloc uşor de determinat când trebuie eliberată conexiunea şi când a fost trimis tot ce era de transmis. S-ar putea avea în vedere un protocol de tipul următor: atunci când 1 termină, trimite ceva de tipul: Am terminat. Ai terminat şi tu? Dacă gazda 2 răspunde: Da, am terminat. Închidem! conexiunea poate fi eliberată în condiţii bune.

Din nefericire, acest protocol nu merge întotdeauna. Binecunoscuta **problemă a celor două armate** este similară acestei situaţii: să ne imaginăm că armată albă şi-a pus tabăra într-o vale (ca în fig. 6-13) Pe amândouă crestele care mărginesc valea sunt armatele albastre. Armata albă este mai mare decât fiecare din cele două armate albastre, dar împreună armatele albastre sunt mai puternice. Dacă oricare din armatele albastre atacă singură, ea va fi înfrântă, dar dacă ele atacă simultan, atunci vor fi victorioase.

Armatele albastre vor să-şi sincronizeze atacul. Totuşi singura lor posibilitate de comunicaţie este să trimită un mesager care să străbată valea. Mesagerul poate fi capturat de armata albă şi mesajul poate fi pierdut (adică vor trebui să utilizeze un canal de comunicaţie nesigur). Problema este următoarea: există vreun protocol care să permită armatelor albastre să învingă?

Să presupunem că comandantul primei armate albastre trimite un mesaj: „Propun să atacăm pe 29 martie”, mesajul ajunge la armata 2 al cărei comandant răspunde: „De acord” iar răspunsul ajunge înapoi la armata 1. Va avea loc atacul în acest caz? Probabil că nu, deoarece comandantul armatei 2 nu ştie dacă răspunsul său a ajuns sau nu la destinaţie. Dacă nu a ajuns, armata 1 nu va ataca, deci ar fi o

prostie din partea lui să intre în luptă.

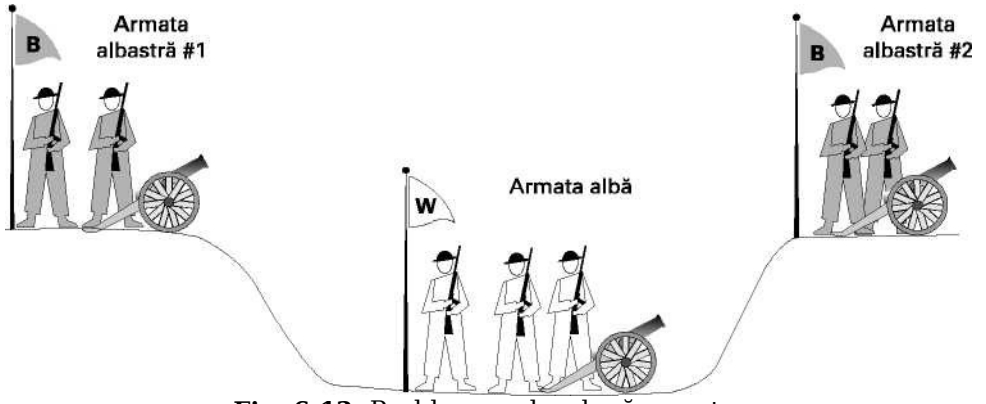


Fig. 6-13. Problema celor două armate.

Să încercăm să îmbunătăţim protocolul, transformându-l într-unul cu înţelegere în trei paşi. Iniţiatorul propunerii de atac trebuie să confirme răspunsul. Presupunând că nici un mesaj nu este pierdut, armata 2 va avea confirmarea, dar comandantul armatei 1 va ezita acum. Până la urmă, el nu ştie dacă confirmarea sa a ajuns la destinaţie şi este sigur că dacă aceasta nu a ajuns, armata 2 nu va ataca. Am putea să încercăm un protocol cu confirmare în patru timpi, dar ne-am lovi de aceleaşi probleme.

De fapt, poate fi demonstrat că nu există un protocol care să funcţioneze. Să presupunem că ar exista un asemenea protocol: decizia finală poate să depindă sau nu de ultimul mesaj al unui asemenea protocol. Dacă nu depinde, putem elimina acest mesaj (şi oricare altul la fel) până ajungem la un protocol în care orice mesaj este vital. Ce se va întâmpla dacă ultimul mesaj este interceptat? Tocmai am hotărât că acest mesaj era unul vital, deci dacă este pierdut, atacul nu va avea loc. Deoarece cel care trimite ultimul mesaj nu poate fi niciodată sigur că mesajul a ajuns, el nu va risca atacând. Mai rău chiar, cealaltă armată albastră ştie şi ea acest lucru, deci nu va ataca nici ea.

Pentru a vedea legătura problemei celor două armate cu problema eliberării conexiunii este suficient să înlocuim ‘atac’ cu ‘deconectare’. Dacă niciuna din părţi nu se deconectează până nu este sigură că cealaltă parte este gata să se deconecteze la rândul ei, atunci deconectarea nu va mai avea loc niciodată.

În practică suntem dispuşi să ne asumăm mai multe riscuri atunci când este vorba de eliberarea conexiunii decât atunci când este vorba de atacarea armatei albe, aşa încât situaţia nu este întru totul fără speranţă. Fig. 6-14 prezintă patru scenarii de eliberare a conexiunii folosind un protocol cu confirmare în trei timpi. Deşi acest protocol nu este infailibil, el este în general adecvat.

În fig. 6-14(a) apare cazul normal în care unul dintre utilizatori trimite un TPDU de tip DR (DISCONNECT REQUEST) pentru a iniţia eliberarea conexiunii. Atunci când acesta soseşte, receptorul trimite înapoi tot un TPDU DR şi porneşte un ceas pentru a trata cazul în care mesajul său este pierdut. Când primeşte mesajul înapoi, iniţiatorul trimite o confirmare şi eliberează conexiunea. În sfârşit, la primirea

confirmării, receptorul eliberează şi el conexiunea. Eliberarea conexiunii înseamnă de fapt că entitatea de transport şterge din tabelele sale informaţia despre conexiunea respectivă din tabela de conexiuni deschise în momentul curent şi semnalează acest lucru utilizatorului nivelului transport. Această acţiune nu este acelaşi lucru cu apelul unei primitive DISCONNECT de către un utilizator al nivelului transport.

Dacă ultima confirmare este pierdută, ca în fig. 6-14(b), putem salva situaţia cu ajutorul ceasului: după scurgerea unui anumit interval de timp conexiunea este eliberată oricum.

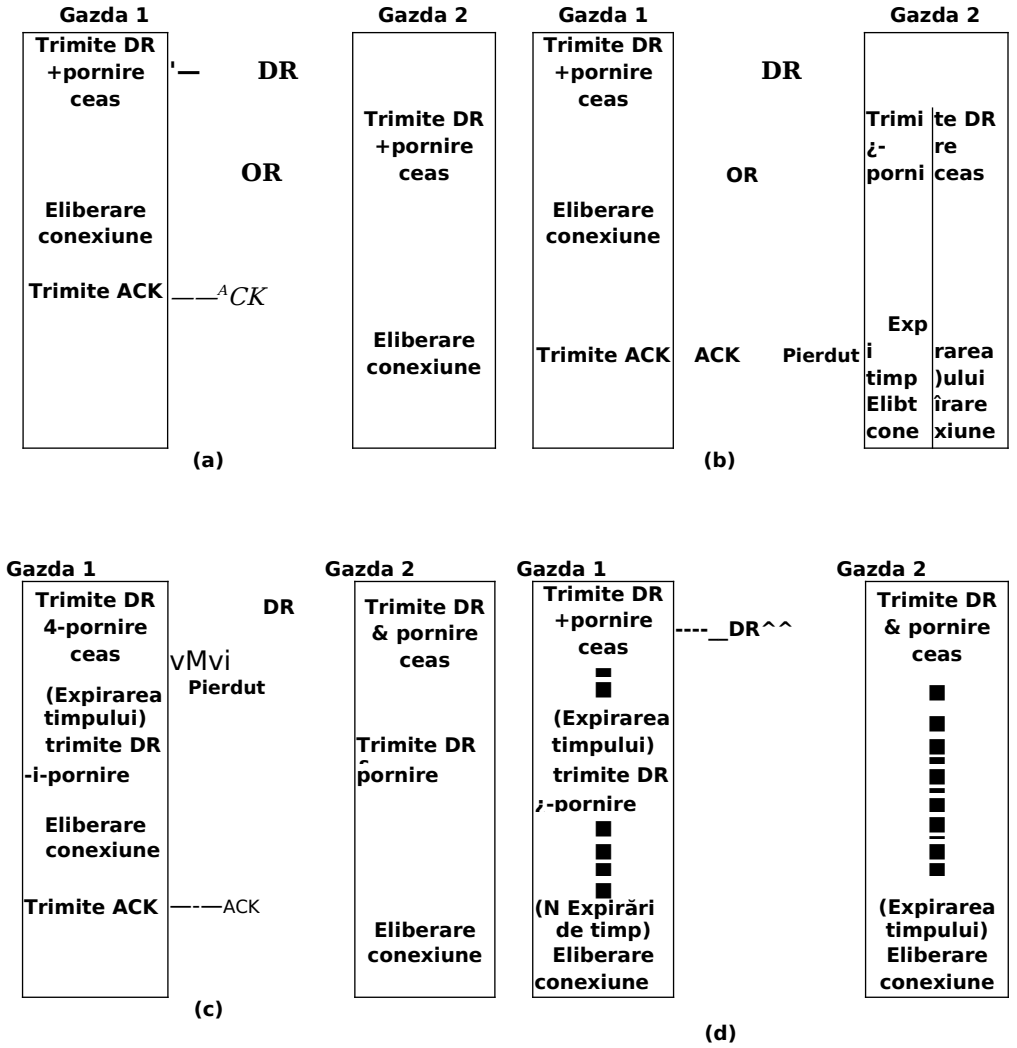


Fig. 6-14. Patru cazuri posibile la eliberarea conexiunii: (a)Cazul normal cu confirmare în trei timpi.
(b)Ultima confirmare este pierdută. (c) Răspunsul este pierdut.
(d) Răspunsul şi următoarele cereri de deconectare sunt pierdute.
(DR=DISCONNECT REQUEST).

Să considerăm acum cazul în care cel de-al doilea DR este pierdut: utilizatorul care a iniţiat deconectarea nu va primi răspunsul aşteptat, va aştepta un anumit timp şi va trimite din nou un DR. În fig. 6-14(c), putem vedea cum se petrec lucrurile în acest caz, presupunând că la a doua încercare toate TPDU-urile ajung corect şi la timp.

Ultima posibilitate pe care o studiem, prezentată în fig. 6-14(d), este similară cu cea din 6-14(c), cu următoarea diferenţă: de aceasta dată niciuna din încercările următoare de a retransmite DR nu reuşeşte. După N încercări, emiţătorul va elibera

pur şi simplu conexiunea. În acelaşi timp, şi receptorul va elibera conexiunea după expirarea timpului.

Deşi acest protocol este, în general, destul de bun, în teorie el poate să dea greş dacă atât mesajul DR iniţial cât şi *N* retransmisii ale sale se pierd. Emiţătorul va renunţa şi va elibera conexiunea, în timp ce la celălalt capăt nu se ştie nimic despre încercările de deconectare şi aceasta va rămâne în continuare activă. În această situaţie rezultă o conexiune deschisă pe jumătate.

Am putea evita această problemă nepermiţând emiţătorului să cedeze după *N* reîncercări nereuşite, ci cerându-i să continue până primeşte un răspuns. Totuşi, dacă celeilalte părţi i se permite să elibereze conexiunea după un interval de timp, este posibil ca iniţiatorul să ajungă să aştepte la infinit. Dacă însă nu s-ar permite eliberarea conexiunii după expirarea unui interval de timp, atunci în cazul din fig. 6-14 (b) protocolul s-ar bloca.

O altă posibilitate de a scăpa de conexiunile pe jumătate deschise este de a aplica o regulă de tipul: dacă nici un TPDU nu soseşte într-un anumit interval de timp, atunci conexiunea este eliberată automat. În acest fel, dacă una din părţi se deconectează, cealaltă parte va detecta lipsa de activitate şi se va deconecta şi ea. Desigur, pentru a implementa aceasta regulă este nevoie ca fiecare entitate de transport să aibă un ceas care va fi repornit la trimiterea oricărui TPDU. La expirarea timpului, se transmite un TPDU vid, doar pentru a menţine conexiunea deschisă. Pe de altă parte, dacă este aleasă această soluţie, şi câteva TPDU-uri vide sunt pierdute la rând pe o conexiune altfel liberă, este posibil ca, mai întâi una din părţi, apoi cealaltă să se deconecteze automat.

Nu vom mai continua să detaliem acest subiect, dar probabil că acum este clar că eliberarea unei conexiuni fără pierderi de date nu este atât de simplă cum părea la început.

6.2.4 Controlul fluxului şi memorarea temporară (buffering)

După ce am studiat în detaliu stabilirea şi eliberarea conexiunii, vom arunca o privire asupra modului în care sunt tratate conexiunile cât timp sunt utilizate. Una din problemele cheie a apărut şi până acum: controlul fluxului. La nivel transport există asemănări cu problema controlului fluxului la nivel legătură de date, dar există şi deosebiri. Principala asemănare: la ambele niveluri este necesar un mecanism (fereastră glisantă sau altceva) pentru a împiedica un emiţător prea rapid să depăşească capacitatea de recepţie a unui receptor prea lent. Principala deosebire: un ruter are în general puţine linii, dar poate să aibă numeroase conexiuni. Această diferenţă face nepractică implementarea la nivel transport a strategiei de memorare temporară a mesajelor folosită la nivel legătură de date.

În protocoalele pentru legătura de date prezentate în cap. 3, cadrele sunt memorate temporar atât de ruterul care emite cât şi de cel care recepţionează. În protocolul 6, de exemplu, atât emiţătorul cât şi receptorul au alocate un număr de *MAXSEQ+1* tampoane pentru fiecare linie, jumătate pentru intrări şi jumătate pentru ieşiri. Pentru un calculator gazdă cu, să spunem, 64 de conexiuni şi numere

de secvenţă de 4 biţi, acest protocol ar necesita 1024 tampoane.

La nivel legătură de date, emiţătorul trebuie să memoreze cadrele transmise, pentru că poate fi necesară retransmiterea acestora. Dacă subreţeaua oferă un serviciu datagramă, atunci entitatea de transport emiţătoare va trebui să memoreze pachetele trimise din aceleaşi motive. Dacă receptorul ştie că emiţătorul stochează toate TPDU-urile până când acestea sunt confirmate, el poate să aloce sau nu tampoane specifice fiecărei conexiuni, după cum i se pare mai bine.

Receptorul poate, de exemplu, să rezerve un singur grup de tampoane pentru toate conexiunile. La sosirea unui TPDU se face o încercare de a obţine dinamic un nou tampon. Dacă un tampon este liber, atunci TPDU-ul este acceptat, altfel, este refuzat. Cum emiţătorul este gata să retransmită TPDU-urile pierdute de subreţea, faptul că unele TPDU-uri sunt refuzate nu produce nici o daună, deşi în acest fel sunt risipite resurse. Emiţătorul va retransmite până când va primi confirmarea.

Pe scurt, dacă serviciul reţea nu este sigur, emiţătorul va trebui să memoreze toate TPDU-urile trimise, la fel ca la nivel legătură de date. Totuşi, folosind un serviciu la nivel reţea sigur sunt posibile unele compromisuri. În particular, dacă emiţătorul ştie că receptorul are întotdeauna tampoane disponibile, atunci nu trebuie să păstreze copiile TPDU-urilor trimise. Totuşi, dacă receptorul nu poate garanta că orice TPDU primit va fi acceptat, emiţătorul va trebui să păstreze copii. În ultimul caz, emiţătorul nu poate avea încredere în confirmarea primită la nivel reţea, deoarece aceasta confirmă sosirea TPDU-ului la destinaţie, dar nu şi acceptarea lui. Vom reveni asupra acestui punct important mai târziu.

Chiar dacă receptorul va realiza memorarea temporară a mesajelor primite, mai rămâne problema dimensiunii tamponului. Dacă cea mai mare parte a TPDU-urilor au aceeaşi dimensiune, este naturală organizarea tampoanelor într-o resursă comună care conţine tampoane de aceeaşi dimensiune, cu un TPDU per tampon, ca în fig. 6-15(a). Dacă însă dimensiunea TPDU-urilor variază de la câteva caractere tipărite la un terminal, la mii de caractere pentru un transfer de fişiere, organizarea ca o resursă comună cu tampoane de aceeaşi dimensiune va pune probleme. Dacă dimensiunea tampoanelor ar fi constantă, egală cu cel mai mare TPDU posibil, atunci va apărea o risipă de spaţiu ori de câte ori este primit un TPDU mai scurt. Dacă dimensiunea tampoanelor este aleasă mai mică decât cel mai mare TPDU posibil, atunci pentru memorarea unui TPDU mai lung vor fi necesare mai multe tampoane, iar complexitatea operaţiei va creşte.

O altă soluţie este utilizarea unor tampoane de dimensiune variabilă, ca în fig. 6-15(b). Avantajul este o mai bună utilizare a memoriei, cu preţul unei gestiuni a tampoanelor mai complicată. O a treia posibilitate este alocarea unui singur tampon circular pentru fiecare conexiune, ca în fig. 6- 15(c). Această soluţie are de asemenea avantajul unei utilizări eficiente a memoriei, dar numai în situaţia în care conexiunile sunt relativ încărcate.

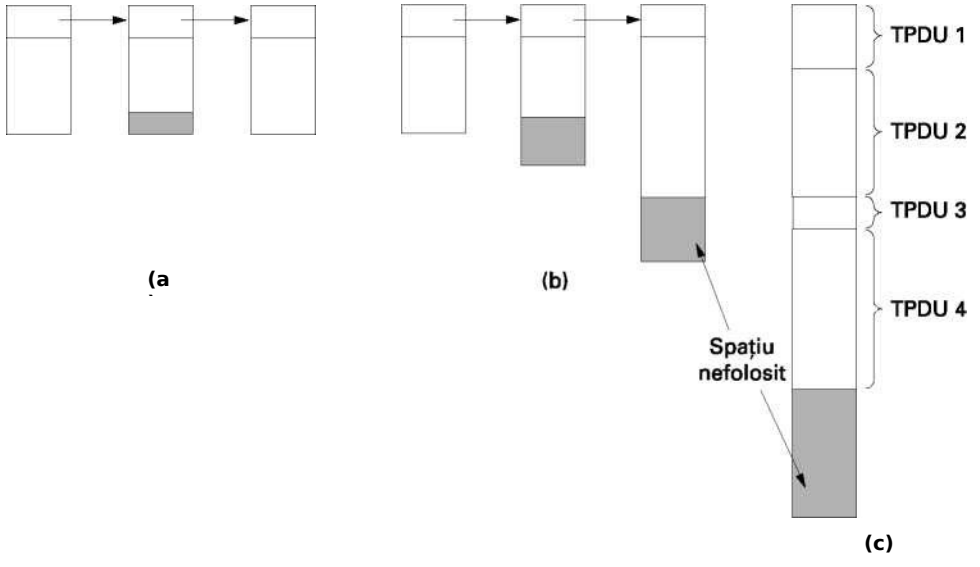


Fig. 6-15. (a) Tampoane de dimensiune fixă înlănţuite. (b) Tampoane de dimensiune variabilă înlănţuite. (c) Un tampon

Compromisul optim între memorarea temporară la sursă sau la destinaţie depinde de tipul traficului prin conexiune. Pentru un trafic în rafală cu o bandă de transfer îngustă, ca traficul produs de

un terminal interactiv, este mai bine ca tampoanele să nu fie prealocate, ci mai curând, alocate dinamic. Întrucât emiţătorul nu poate să fie sigur că receptorul va reuşi să aloce un tampon la sosirea unui pachet, emiţătorul va fi nevoit să reţină copia unui TPDU transmis până când acesta va fi confirmat. Pe de altă parte, pentru un transfer de fişiere sau pentru orice alt trafic care necesită o bandă de transfer largă este mai bine dacă receptorul alocă un set întreg de tampoane, pentru a permite un flux de date la viteză maximă. Cu alte cuvinte, pentru un trafic în rafală cu o bandă de transfer îngustă este mai bine să fie folosite tampoane la emiţător, în timp ce pentru un trafic continuu cu o band ă de transfer largă, este mai eficientă folosirea tampoanelor la receptor.

Pe măsură ce conexiunile sunt create şi eliberate de trafic, iar şablonul se schimbă, emiţătorul şi receptorul trebuie să îşi ajusteze dinamic politica de alocare a tampoanelor. În consecinţă, protocolul de transport trebuie să permită emiţătorului să ceară spaţiu pentru tampoane la capătul celălalt al conexiunii. Tampoanele pot fi alocate pentru o anumită conexiune sau pot fi comune pentru toate conexiunile între două calculatoare gazdă. O alternativă este ca receptorul, cunoscând situaţia tampoanelor sale (dar necunoscând şablonul traficului) să poată spune emiţătorului: „Am rezervat *X* tampoane pentru tine”. Dacă numărul conexiunilor deschise trebuie să crească, poate fi necesar ca spaţiul alocat unei singure conexiuni să scadă, deci protocolul trebuie să furnizeze şi această facilitate.

O modalitate înţeleaptă de a trata alocarea dinamică a tampoanelor este separarea stocării în tampoane de confirmarea mesajelor, spre deosebire de protocolul cu fereastră glisantă din cap. 3. Alocarea dinamică a tampoanelor înseamnă, de fapt, o fereastră cu dimensiune variabilă. La început, emiţătorul trimite cereri pentru un anumit număr de tampoane bazându-se pe o estimare a necesităţilor. Receptorul îi alocă atâtea tampoane cât îşi poate permite. De fiecare dată când emiţătorul trimite un TPDU, el decrementează numărul de tampoane pe care le are alocate la receptor, oprindu-se când acest număr devine zero. Receptorul trimite înapoi confirmări şi situaţia tampoanelor alocate, împreună cu traficul în sens invers.

Fig. 6-16 este un exemplu pentru modul în care administrarea dinamică a ferestrelor poate fi folosită într-o subreţea cu datagrame, cu numere de secvenţă pe 4 biţi. Să presupunem că informaţia despre alocarea tampoanelor este împachetată în TPDU-uri distincte şi că este separată de traficul în sens invers. La început A doreşte opt tampoane, dar nu i se acordă decât patru. După aceea trimite trei TPDU-uri, iar al treilea este pierdut. TPDU-ul 6 confirmă recepţia tuturor TPDU-urilor cu numere de secvenţă mai mici sau egale cu 1, permiţând lui A să elibereze acele tampoane şi, mai mult, îl informează pe A că B poate să mai recepţioneze trei TPDU-uri (adică TPDU-urile cu numere de secvenţă 2, 3, 4). A ştie că a trimis deja numărul 2, deci se gândeşte că ar putea trimite 3 şi 4, ceea ce şi încearcă să facă. În acest moment, el este blocat şi trebuie să aştepte alocarea unui tampon. Expirarea timpului pentru primirea confirmării determină retransmisia mesajului 2 (linia 9), care poate avea loc, deşi emiţătorul este blocat, deoarece se vor utiliza tampoane deja alocate. În linia 10, B confirmă primirea tuturor TPDU-urilor până la 4, dar îl ţine pe A încă blocat. O astfel de situaţie ar fi fost imposibilă cu protocolul

cu fereastră glisantă prezentat în cap. 3. Următorul TPDU de la B la A alocă încă un tampon şi îi permite lui A să continue.

În cazul strategiilor pentru alocarea tampoanelor de tipul celei de mai sus, eventuale probleme pot să apară în reţele neorientate pe conexiune dacă sunt pierdute TPDU-uri de control. Să privim linia 16 din fig. 6-16: B a mai alocat tampoane pentru A, dar TPDU-ul care transmitea această informaţie a fost pierdut. Deoarece TPDU-urile de control nu sunt numerotate sau retransmise, A va fi blocat. Pentru a preveni această situaţie, fiecare calculator gazdă trebuie să trimită periodic pe fiecare conexiune TPDU-uri de control ce pot conţine confirmări şi informaţii despre starea tampoanelor. În acest fel, A va fi deblocat mai devreme sau mai târziu.

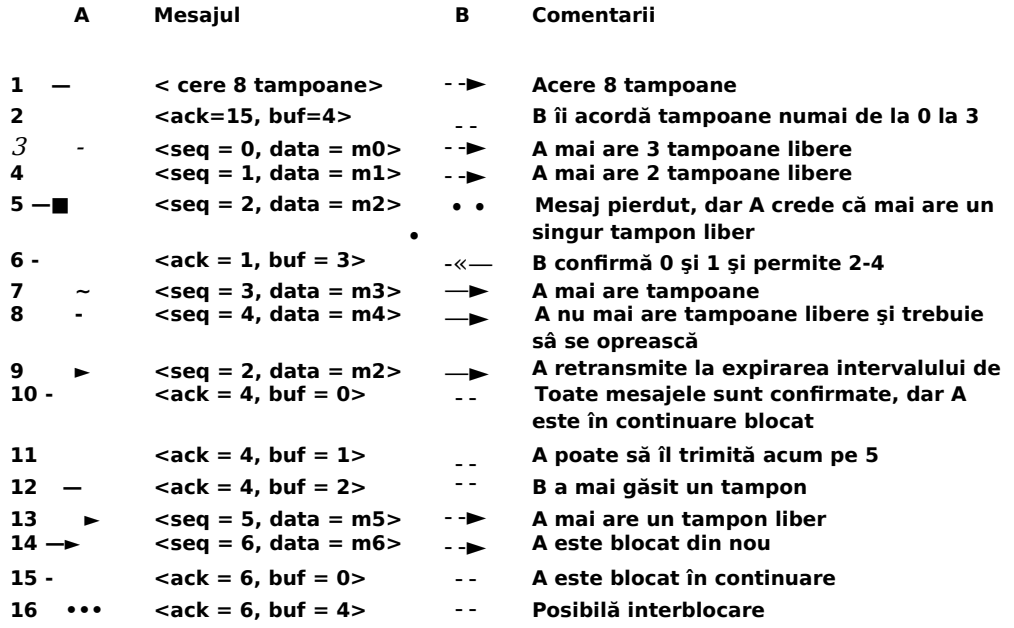


Fig. 6-16. Alocarea dinamică a tampoanelor. Săgeţile indică direcţia transmisiei. Punctele de suspensie (...) indică pierderea unui TPDU.

Până acum am presupus tacit că singura limită impusă ratei de transfer a emiţătorului este legată de dimensiunea spaţiului alocat la receptor pentru tampoane. Deoarece preţul memoriei continuă să scadă vertiginos, ar putea deveni posibilă echiparea unei gazde cu suficient de multă memorie, astfel încât lipsa tampoanelor să pună rar probleme, dacă le va pune vreodată.

Atunci când spaţiul de memorie alocat pentru tampoane nu limitează fluxul maxim, va apărea o altă limitare: capacitatea de transport a subreţelei. Dacă două rutere adiacente pot să schimbe cel mult x pachete pe secundă şi există k căi distincte între două calculatoare gazdă, atunci este imposibil transferul la o rată mai mare de $k * x$ TPDU-uri pe secundă, oricât de multe tampoane ar fi alocate la cele două capete ale conexiunii. Dacă emiţătorul se grăbeşte prea tare (adică trimite mai mult de $k * x$ TPDU-uri pe secundă), reţeaua se va congestiona,

deoarece nu va putea să livreze datele le fel de repede cum le primeşte.

Este necesar un mecanism bazat pe capacitatea de transport a subreţelei şi nu pe capacitatea de memorare în tampoane a receptorului. Evident, mecanismul de control al fluxului trebuie aplicat la emiţător pentru a preveni existenţa prea multor TPDU-uri neconfirmate la acesta. Belsens (1975) a propus folosirea unei scheme cu fereastră glisantă în care emiţătorul modifică dinamic dimensiunea ferestrei pentru a o potrivi la capacitatea de transport a reţelei. Dacă reţeaua poate să transporte *c* TPDU-uri pe secundă şi durata unui ciclu (incluzând transmisia, propagarea, timpul petrecut în cozi, prelucrarea la destinaţie şi revenirea confirmării) este *r,* atunci dimensiunea ferestrei la emiţător trebuie să fie $c * r$. Folosind o fereastră cu această dimensiune, emiţătorul va putea lucra la capacitate maximă. Orice mică scădere a performanţelor reţelei va genera blocări ale emiţătorului.

Pentru a ajusta periodic dimensiunea ferestrei, emiţătorul poate urmări cei doi parametri, după care poate calcula dimensiunea dorită a ferestrei. Capacitatea de transport a subreţelei poate fi determinată pur şi simplu numărând TPDU-urile confirmate într-o anumită perioadă de timp şi raportând la acea perioadă. În timpul măsurătorii, emiţătorul trebuie să transmită cât mai repede pentru a fi sigur că factorul care limitează numărul confirmărilor este capacitatea de transport a subreţelei şi nu rata mică de emisie. Timpul necesar pentru ca un TPDU transmis să fie confirmat poate fi măsurat cu exactitate şi media poate fi calculată continuu. Deoarece capacitatea de transport a reţelei disponibilă pentru orice flux dat variază în timp, dimensiunea ferestrei trebuie ajustată frecvent pentru a urmări schimbările în capacitatea de transport. Aşa cum vom vedea mai departe, în Internet se foloseşte un mecanism similar.

6.2.5 Multiplexarea

Multiplexarea mai multor conversaţii pe conexiuni, circuite virtuale şi legături fizice joacă un rol important în mai multe niveluri ale arhitecturii reţelei. În cazul nivelului transport, multiplexarea poate fi necesară din mai multe motive. De exemplu, dacă doar o singură adresă de reţea este disponibilă pe o gazdă, toate conexiunile transport de pe acea maşină trebuie să o folosească. Când un TDPU soseşte este necesar un mod de a spune cărui proces trebuie dat. Această situaţie numită **multiplexare în sus**, este prezentată în fig. 6-17(a). În această figură, patru conexiuni transport diferite folosesc în comun aceeaşi conexiune reţea (de exemplu, adresa IP) către calculatorul gazdă de la distanţă.

Adresă la nivel

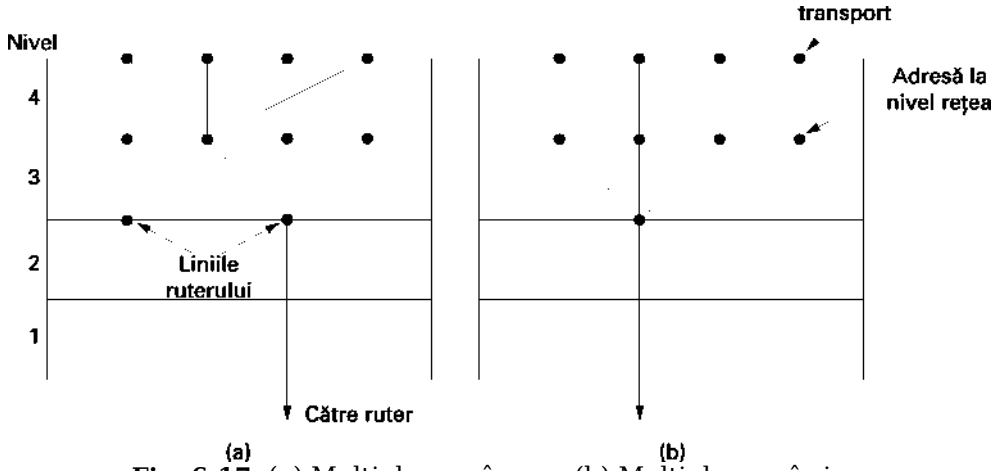


Fig. 6-17. (a) Multiplexare în sus. (b) Multiplexare în jos.

Multiplexarea poate să fie utilă nivelului transport şi din alt motiv, legat de deciziile tehnice şi nu de politica de preţuri ca până acum. Să presupunem, de exemplu, că o subreţea foloseşte intern circuite virtuale şi impune rată de date maximă p fiecare dintre ele. Dacă un utilizator are nevoie de mai multă lăţime de bandă decât poate oferi un circuit virtual, o soluţie este ca nivelul transport să deschidă mai multe conexiuni reţea şi să distribuie traficul prin acestea (într-un sistem round-robin), la fel ca în fig. 6-17(b). Acest mod de operare se numeşte **multiplexare în jos.** În cazul a *k* conexiuni reţea deschise lăţimea de bandă reală este multiplicată cu un factor *k*. Un exemplu obişnuit de multiplexare în jos apare la utilizatorii care au acasă o linie ISDN. Această linie pune la dispoziţie două conexiuni separate de 64 Kbps. Folosirea ambelor pentru apelul unui distribuitor de Internet şi împărţirea traficului pe ambele linii, face posibilă atingerea unei lăţimi de bandă efectivă de 128 Kbps.

6.2.6 Refacerea după cădere

În cazul în care calculatoarele gazdă sau ruterele se întâmplă să cadă, recuperarea după o astfel de cădere devine o problemă. Dacă entitatea de transport este în întregime conţinută de calculatorul gazdă, atunci revenirea după o cădere a reţelei sau a unui ruter este simplă. Dacă nivelul reţea furnizează un serviciu datagramă, atunci entitatea de transport ştie să rezolve problema TPDU-urilor pierdute. Dacă nivelul reţea furnizează un serviciu orientat pe conexiune, atunci pierderea unui circuit virtual este tratată stabilind un circuit virtual nou şi apoi întrebând entitatea de transport aflată la distanţă care TPDU-uri a primit deja şi ce TPDU-uri nu. Acestea din urmă pot fi retransmise.

Căderea unui calculator gazdă pune o problemă mult mai supărătoare. În particular, clienţii pot dori să continue să lucreze imediat după ce serverul cade şi reporneşte. Pentru a ilustra această dificultate, să presupunem că o gazdă (clientul) trimite un fişier lung unei alte gazde (serverul) folosind un protocol simplu de tip pas-cu-pas (stop-and-wait). Nivelul transport de pe server nu face decât să paseze

utilizatorului TPDU-urile primite, unul câte unul. Dacă în timpul transmisiei serverul cade, la revenirea acestuia tabelele sunt reiniţializate şi serverul nu va mai şti precis unde a rămas.

Într-o încercare de a reveni în starea sa iniţială, serverul ar putea să trimită cereri tuturor celorlalte gazde anunţând că tocmai s-a refăcut după o cădere şi cerând clienţilor să-l informeze despre situaţia tuturor conexiunilor deschise. Fiecare client poate fi în una din următoarele stări: un TPDU neconfirmat (starea S1) sau nici un TPDU neconfirmat (starea S0). Bazându-se numai pe această informaţie, clientul trebuie să decidă dacă să retransmită sau nu cel mai recent TPDU.

La prima vedere pare evident: atunci când află de căderea şi revenirea serverului, clientul trebuie să retransmită doar dacă are TPDU-uri neconfirmate (adică este în starea S1). Totuşi, o privire mai atentă descoperă dificultăţile care apar în această abordare naivă. Să considerăm, ca exemplu, situaţia în care entitatea de transport de pe server trimite mai întâi confirmarea şi apoi, după ce confirmarea a fost trimisă, pasează datele procesului aplicaţie. Trimiterea confirmării şi transferul datelor procesului aplicaţie sunt două evenimente distincte care nu pot avea loc simultan. Dacă o cădere a serverului are loc după trimiterea confirmării, dar înainte de transferul datelor, clientul va primi confirmarea şi se va afla în starea S0, atunci când anunţul despre cădere ajunge la el. În acest caz, clientul nu va mai retransmite (ceea ce este incorect), crezând că TPDU-ul respectiv a fost recepţionat. Această decizie a clientului va conduce la lipsa unui TPDU.

În acest moment s-ar putea spune: „Nimic mai simplu! Tot ceea ce trebuie făcut este să repro- gramăm entitatea de transport astfel, încât să transfere datele mai întâi şi să trimită confirmarea după aceea!”. Să vedem: dacă transferul datelor a avut loc, dar serverul cade înainte să trimită confirmarea, clientul va fi în starea S1, va retransmite şi astfel vom avea un TPDU duplicat în fluxul de date către procesul aplicaţie.

Oricum ar fi proiectaţi clientul şi serverul, întotdeauna vor exista situaţii în care revenirea după o cădere nu se va face corect. Serverul poate fi programat în două feluri: să facă mai întâi confirmarea sau să transfere mai întâi datele. Clientul poate fi programat în patru feluri: să retransmită întotdeauna ultimul TPDU, să nu retransmită niciodată, să retransmită numai în starea S0, să retransmi- tă numai în starea S1. Rezultă astfel opt combinaţii, dar, aşa cum vom vedea, pentru fiecare combinaţie există o anumită succesiune de evenimente pentru care protocolul nu funcţionează corect.

La server sunt posibile trei evenimente: trimiterea unei confirmări (A), transferul datelor la procesul aplicaţie (T) şi căderea (C). Aceste trei evenimente pot să fie ordonate în 6 feluri: AC(T), ATC, C(AT), C(TA), TAC şi TC(A). Parantezele sunt folosite pentru a indica faptul că nici A, nici

T nu pot urma după C (odată ce serverul a căzut, e bun căzut). Fig. 6-18 prezintă toate cele opt combinaţii ale strategiilor clientului şi serverului şi prezintă secvenţele valide pentru fiecare din ele. Se observă că pentru orice strategie există o secvenţă de evenimente pentru care protocolul nu funcţionează corect. De exemplu, dacă clientul retransmite întotdeauna, secvenţa ATC va genera un duplicat care nu poate fi detectat, în timp ce pentru celelalte două secvenţe totul este în regulă.

OK = Protocolul funcţionează corect **Mai întâi confirmă, apoi** **Mai întâi transferă, apoi**
DUP = Protocolul generează un mesaj duplicat LOST = Protocol
pierde un mesaj

Strategia folosită de emiţător	Strategia folosită de receptor		
	AC(T)	ATC	C(AT
Retransmite întotdeauna	OK	DUP	OK
Nu retransmite	LOST	OK	LOST
Retransmite în	OK	DUP	LOST
Retransmite în	LOST	OK	OK

	C(TA)	T AC	TC(A)
	OK	DUP	DUP
	LOST	OK	OK
	LOST	DUP	OK
	OK	OK	DUP

Fig. 6-18. Combinaţiile [illegible] posibi[illegible].

Încercarea de a reproiect[illegible]colu[illegible]ientul şi serverul schimbă mai mu[illegible] ca[illegible]atele, astfel încât clientul ştie exa[illegible]e p[illegible] dacă serverul a căzut imediat înainte sau imediat după tra[illegible]zia se impune de la sine: dată fiind condiţia de bază ca două evenimente să nu fie simultane, revenirea după o cădere nu poate fi făcută transparent pentru nivelurile superioare.

În termeni mai generali, acest rezultat poate fi reformulat astfel: restartarea după o cădere a nivelului N nu poate fi făcută decât de către nivelul N+1, şi aceasta doar dacă nivelul superior reţine suficientă informaţie de stare. Aşa cum am arătat mai sus, nivelul transport poate să revină după erorile nivelului reţea numai dacă fiecare capăt al conexiunii ţine minte unde a rămas.

Am ajuns astfel la problema definirii precise a ceea ce înseamnă o confirmare capăt-la-capăt. În principiu, protocolul de transport este unul capăt-la-capăt şi nu înlănţuit ca la nivelurile de mai jos. Să considerăm cazul unui utilizator care generează cereri de tranzacţii pentru o bază de date de la distanţă. Să presupunem că entitatea de transport aflată la distanţă este programată să trimită mai întâi TPDU-ul nivelului superior şi apoi să confirme. Chiar şi în acest caz, primirea unei confirmări de către maşina utilizator nu înseamnă neapărat că maşina gazdă de la distanţă a avut timp să actualizeze baza de date. De fapt, o adevărată confirmare capăt-la-capăt, a cărei primire arată că s-au efectuat toate prelucrările de către

maşina de la distanţă, este probabil imposibil de obţinut. Acest subiect este discutat în detaliu de Saltser ş.a. (1984).

6.3 UN PROTOCOL SIMPLU DE TRANSPORT

Pentru a concretiza ideile discutate, în această secţiune vom studia în detaliu un exemplu de nivel transport. Vom utiliza setul abstract de primitive orientate pe conexiune prezentat în fig. 6-2. Alegerea acestor primitive orientate pe conexiune face exemplul ales similar cu (dar mai simplu decât) popularul protocol TCP.

6.3.1 Primitivele serviciului ales ca exemplu

Prima problemă constă în definirea exactă a primitivelor de transport. Pentru CONNECT este uşor: vom avea o rutină de bibliotecă *connect* care poate fi apelată cu parametri potriviţi pentru stabilirea unei conexiuni. Parametrii sunt TSAP-ul local şi cel aflat la distanţă. În timpul apelului, apelantul este blocat în timp ce entitatea de transport încearcă să stabilească conexiunea. Dacă conexiunea reuşeşte, apelantul este deblocat şi poate să înceapă să transmită date.

Atunci când un proces doreşte să accepte conexiuni, el face un apel *listen* specificând un anumit TSAP pe care îl ascultă. După aceasta, procesul este blocat până când un proces aflat la distanţă încearcă să stabilească o conexiune cu TSAP-ul la care aşteaptă.

De observat că acest model este asimetric. O parte este pasivă, executând *listen* şi aşteptând ca ceva să se întâmple. Cealaltă parte este activă şi iniţiază conexiunea. O întrebare interesantă este: ce trebuie făcut dacă partea activă începe prima? O posibilitate este ca încercarea de conectare să nu reuşească dacă nu există nici un proces care să asculte la TSAP-ul aflat la distanţă. O altă posibilitate este ca iniţiatorul să se blocheze (eventual pentru totdeauna) până când apare un proces care să asculte la TSAP-ul aflat la distanţă.

Un compromis folosit în exemplul nostru este păstrarea cererii de conexiune la receptor pentru un anumit interval de timp. Dacă un proces de pe acest calculator gazdă apelează *listen* înainte ca timpul să expire, atunci conexiunea este stabilită, altfel conexiunea este refuzată şi apelantul este deblocat întorcând un cod de eroare.

Pentru a elibera o conexiune vom folosi un apel *disconnect.* Atunci când ambele părţi s-au deconectat, conexiunea este eliberată. Cu alte cuvinte, folosim un model de deconectare simetric.

Transmisia de date are exact aceeaşi problemă ca şi stabilirea conexiunii: emiţătorul este activ, dar receptorul este pasiv. Vom folosi aceeaşi soluţie pentru transmisia de date ca şi pentru stabilirea conexiunii, adică un apel activ *send* care trimite datele şi un apel pasiv *receive* care blochează până când soseşte un TPDU.

Definiţia concretă a serviciului constă astfel din cinci primitive: CONNECT, LISTEN, DISCONNECT, SEND şi RECEIVE. Fiecare primitivă corespunde unei

funcţii de bibliotecă care execută acea primitivă. Parametrii pentru primitive şi funcţiile de bibliotecă sunt:

connum = LISTEN(local)
connum = CONNECT (local, remote)
status = SEND(connum, buffer, bytes) status = RECEIVE(connum, buffer, bytes) status = DISCONNECT(connum)

Primitiva LISTEN anunţă disponibilitatea serverului de a accepta cereri de conexiune la TSAP- ul indicat. Utilizatorul primitivei este blocat până când se face o încercare de conectare la TSAP-ul specificat. Blocarea poate să fie definitivă.

Primitiva CONNECT are doi parametri, un TSAP local (adică o adresă la nivel transport) şi un TSAP aflat la distanţă şi încearcă să stabilească o conexiune între acestea două. Dacă reuşeşte, ea întoarce *connum,* un număr nenegativ utilizat pentru a identifica conexiunea. Dacă nu reuşeşte, motivul este pus în *connum* ca un număr negativ. În modelul nostru (destul de simplu), fiecare TSAP poate să participe la cel mult o singură conexiune de transport, deci un motiv pentru refuzul unei cereri de conectare poate fi că adresa la nivel transport este deja folosită. Alte câteva motive pot fi: gazda de la distanţă căzută, adresă locală incorectă sau adresă de la distanţă incorectă.

Primitiva SEND trimite conţinutul unui tampon ca un mesaj pe conexiunea indicată, eventual divizându-l în mai multe unităţi, dacă este nevoie. Erorile posibile, întoarse în *status,* pot fi: nu există conexiune, adresă de tampon invalidă sau număr de biţi negativ.

Primitiva RECEIVE indică faptul că apelantul aşteaptă să recepţioneze date. Dimensiunea mesajului este plasată în câmpul *bytes.* Dacă procesul aflat la distanţă a eliberat conexiunea sau dacă adresa pentru tampon este incorectă (de exemplu, în afara spaţiului de adrese al programului utilizator), funcţia va întoarce un cod de eroare indicând natura problemei.

Primitiva DISCONNECT pune capăt unei conexiunii transport indicate prin parametrul *connum*. Erori posibile sunt: *connum* aparţine de fapt altui proces sau *connum* nu este un identificator valid de conexiune. Codul de eroare sau 0 pentru succes sunt returnate în *status.*

6.3.2 Entitatea de transport aleasă ca exemplu

Înainte de a studia programul aferent entităţii de transport, trebuie spus că acesta este un exemplu similar celor din cap. 3: este prezentat mai mult în scop pedagogic decât pentru a fi utilizat. Mai multe detalii tehnice (precum detectarea extensivă a erorilor) care ar fi necesare unui produs real au fost lăsate deoparte pentru simplitate.

Nivelul transport utilizează primitivele serviciului reţea pentru a trimite şi recepţiona TPDU-uri. Trebuie să alegem primitivele serviciului reţea pe care le vom utiliza pentru acest exemplu. O posibilitate ar fi fost: un serviciu datagramă nesigur. Pentru a păstra simplitatea exemplului, nu am făcut această alegere. Folosind un serviciu datagramă nesigur, codul pentru entitatea de transport ar fi

devenit foarte mare şi complicat, în cea mai mare parte legat de pachete pierdute sau întârziate. Şi în plus, cea mai mare parte a acestor idei au fost deja discutate în cap. 3.

Am ales în schimb un serviciu reţea sigur, orientat pe conexiune. În acest fel ne putem concentra asupra problemelor puse de nivelul transport care nu apar în nivelurile inferioare. Acestea includ, între altele, stabilirea conexiunii, eliberarea conexiunii şi gestiunea tampoanelor. Un serviciu de transport simplu, construit peste un nivel reţea ATM, ar putea să semene cu acesta.

În general, entitatea de transport poate să fie parte a sistemului de operare al calculatorului gazdă sau poate să fie un pachet de funcţii de bibliotecă în spaţiul de adrese utilizator. Pentru simplitate, exemplul nostru a fost programat ca şi cum entitatea de transport ar fi un pachet de funcţii de bibliotecă, dar schimbările necesare pentru a o face parte a sistemului de operare sunt minime (fiind în principal legate de modul cum sunt adresate tampoanele).

Totuşi merită remarcat faptul că, în acest exemplu, „entitatea de transport” nu este o entitate separată, ci este o parte a procesului utilizator. Mai precis, atunci când utilizatorul foloseşte o primitivă blocantă, de exemplu LISTEN, se blochează toată entitatea de transport. În timp ce această arhitectură este potrivită pentru un calculator gazdă cu un singur proces utilizator, pentru un calculator gazdă cu mai mulţi utilizatori este normal ca entitatea de transport să fie un proces separat, distinct de toate celelalte procese.

Interfaţa cu nivelul reţea se face prin intermediul procedurilor *to_net* şi *from_net.* Fiecare are şase parametri. Primul este identificatorul conexiunii, care este în corespondenţă bijectivă cu circuitul virtual la nivel reţea. Apoi urmează biţii *Q* şi *M* care, atunci când au valoarea 1, indică mesaje de control şi, respectiv, faptul că mesajul continuă şi în următorul pachet. După acestea urmează tipul pachetului, ales dintr-un set de şase tipuri de pachete prezentate în fig. 6-19. La sfârşit se găseşte un indicator către zona de date şi un întreg care indică numărul de octeţi de date.

Tip pachet	Explicaţii
CALL REQUEST	Trimis pentru a stabili conexiunea
CALL ACCEPTED	Răspuns la CALL REQUEST
CLEAR REQUEST	Trimis pentru a elibera conexiunea
CLEAR CONFIRMATION	Răspuns la CLEAR REQUEST
DATA	Pentru transport de date
CREDIT	Pachet de control pentru gestionarea

Fig. 6-19. Tipurile de pachete folosite la nivel reţea.

La apelurile *to_net,* entitatea de transport completează toţi parametrii pentru ca nivelul reţea să-i poată citi; la apelurile *from_net* nivelul reţea dezasamblează un pachet sosit pentru entitatea de transport. Transferul informaţiei sub forma unor parametri ai unei proceduri şi nu a pachetului de trimis sau de recepţionat protejează nivelul transport de toate detaliile protocolului nivelului reţea. Dacă entitatea de transport încearcă să trimită un pachet atunci când fereastra glisantă corespunzătoare a circuitului virtual este plină, ea va fi blocată într-un apel *to_net* până când va fi spaţiu în fereastră. Mecanismul este transparent pentru entitatea de

transport şi este controlat de nivelul reţea prin comenzi ca *enable_transport_layer* şi *disable_transport_layer,* analoage celor descrise în protocoalele din cap. 3. Gestionarea ferestrei de pachete este de asemenea făcută de către nivelul reţea.

Pe lângă acest mecanism transparent de suspendare mai există două proceduri (care nu sunt prezentate aici), *sleep* şi *wakeup,* apelate de entitatea de transport. Procedura *sleep* este apelată atunci când entitatea de transport este blocată logic în aşteptarea unui eveniment extern, în general sosirea unui pachet. După ce a fost apelat *sleep,* entitatea de transport (şi procesul utilizator, bineînţeles) sunt blocate.

Codul entităţii de transport este prezentat în fig. 6-20. Orice conexiune este într-una din următoarele şapte stări:

1. *IDLE* – conexiunea nu a fost încă stabilită
2. *WAITING* – a fost executat CONNECT şi trimis: CALL_REQUEST
3. *QUEUED* – a sosit un CALL_REQUEST, nu a fost executat încă nici un apel LISTEN
4. *ESTABLISHED* – conexiunea a fost stabilită
5. *SENDING* – utilizatorul aşteaptă permisiunea de a trimite un pachet
6. *RECEIVING* – a fost apelat RECEIVE
7. *DISCONNECTING* – un apel DISCONNECT a fost făcut local

Pot să apară tranziţii între stări ori de câte ori are loc unul din următoarele evenimente: este executată o primitivă, soseşte un pachet sau expiră un interval de timp stabilit.

Procedurile prezentate în fig. 6-20 sunt de două tipuri. Majoritatea sunt apelate direct de către programele utilizator, totuşi *packet_arrival* şi *clock* sunt diferite. Execuţia lor este declanşată de evenimente externe: sosirea unui pachet şi, respectiv, avansul ceasului. Ele sunt de fapt rutine de întrerupere. Vom presupune că acestea nu sunt niciodată apelate atât timp cât rulează o altă procedură a entităţii de transport. Acestea pot fi executate numai atunci când procesul utilizator este în aşteptare sau atât timp cât el rulează în afara entităţii de transport. Această proprietate este crucială pentru funcţionarea corectă a entităţii de transport.

```
 #define MAX_CONN 32                    /* numărul maxim de conexiuni deschise simultan
 #define MAX_MSG_SIZE 8192            */ /* dimensiunea maximă a unui mesaj în octeţi */ /*
 #define MAX_PKT_SIZE 512                 dimensiunea maximă a unui pachet în octeţi */
 #define TIMEOUT 20
 #define CRED 1
 #define OK 0

#define ERR_FULL-1
#define ERR_REJECT -2
#define ERR_CLOSED -3
#define LOW_ERR -3 typedef int transport_address;
typedef enum {CALL_REQ,CALL_ACC, CLEAR_REQ, CLEAR_CONF, DATA_PKT,CREDIT} pkt_type;
typedef enum {IDLE, WAITING, QUEUED, ESTABLISHED, SENDING, RECEIVING, DISCONN}
cstate;

/* Variabile globale */ transport_address listen_address; int listen_conn;
unsigned char data[MAX_PKT_SIZE] struct              /* adresa locală care este ascultată */
conn {                                   /* identificatorul de conexiune pentru listen */
     transport_address local_address,                /* zona pentru pachetele de date */
     remote_address; estate state;     /* starea conexiunii */ /* pointer la tamponul de recepţie
     unsigned char *user_buf_addr;       */ /* contor de emisie/recepţie*/ /* setat atunci când
     int byte_count;                   este primit un pachet CLEAR_REQ*/ /* folosit pentru a
     int clr_req_received;             evita aşteptările infinite */ /* numărul de mesaje care
     int timer;                                poate fi trimis */ /* poziţia 0 nu e folosită */
     int credits;
}; conn[MAX_CONN + 1] /*
prototipuri */
void sleep(void);
void wakeup(void);
void to_net(int cid, int q, int *m, pkt_type pt, unsigned char *p, int bytes);
void from_net(int *cid, int *q, int *m, pkt_type *pt,unsigned char *p, int *bytes);

int listen (transport_address t)
{
/* Utilizatorul vrea stabilească o conexiune. Trebuie văzut daca CALL_REQ a sosit deja */ int i,
     found=0;

     for (i=1 ; i<= MAX_CONN; i++)          /* caută în tabela de conexiuni CALL_REQ*/
          if (conn[i].state == QUEUED && conn[i].locall_address == t) { found = i; break;
          }
                                  /* nu a găsit nici un CALL_REQ. Aşteaptă până când soseşte
               unul sau până când expira intervalul de timp de aşteptare */ listen_address = t;
               sleep(); i = listen_conn;
           }
      conn[i].state = ESTABLISHED;

         if (found == 0) {
                                                          /* conexiunea este stabilită */
     conn[i].timer = 0                                     /* ceasul nu este folosit */
     listen_conn = 0;                              /* 0 este presupus a fi o adresă invalidă */
     to_net(i, 0, 0, CALL_ACC, 0);       /* spune niv. reţea să accepte cererea de conexiune */
     return(i);                                    /* întoarce identificatorul conexiunii */
}

int connect (transport_address l, transport_address r)
{                              /* Utilizatorul doreşte să stabilească o conexiune cu un proces aflat
```

```
                                                    la distanţă; se trimite un pachet CALL_REQ */
    int i;
    struct conn *cptr;

    data[0] = r;                                /* pachetul CALL_REQ are nevoie de acestea */
    data[ 1 ] = l;
    i = MAX_CONN;                                                    /* caută în tabelă înapoi */
    while (conn[i].state != IDLE && i>1) i = i-1;
    if (conn[i].state == IDLE) {                                     /* Face o intrare în tabelă */
        cptr = &conn[i];
        cptr->local_address = l;; cptr->remote_address = r; cptr->state = WAITING; cptr-
        >clr_req_received = 0; cptr->credits = 0; cptr->timer = 0;
        to_net(i, 0, 0, CALL_REQ, data, 2);
        sleep();                                        /* aşteaptă CALL_ACC sau CLEAR_REQ */
        if (cptr->state == ESTABLISHED) return (i);
        if (cptr->clr_req_received ) {                   /* cererea de conexiune este refuzată */
            cptr->state = IDLE                                       /* înapoi în starea IDLE */
            to_net(i, 0, 0, CLEAR_CONF, data, 0);
            return(ERR_REJECT);
        }
    }
    else return (ERR_FULL)          /* conexiunea este refuzată: insuficient spaţiu în tabele */
}

int send (int cid, unsigned char bufptr[], int bytes)
{                                                    /* Utilizatorul doreşte să trimită un mesaj */
    int i, count, m;
    struct conn *cptr = &conn[cid];

    /* Intr ă în starea SENDING */
    cptr->state = SENDING;
    cptr->byte_count = 0;
    if (cptr->clr_req_received == 0 && cptr->credits == 0) sleep();
    if (cptr->clr_req_received == 0) {
                                /* Există credite; împarte mesajul în pachete dacă este cazul */
        do {
            if (bytes – cptr->byte_count > MAX_PKT_SIZE) {
                /* mesaj format din mai multe pachete */
                count = MAX_PKT_SIZE;
                m = 1;
                /* mai urmează pachete */

            } else { /* un mesaj format dintr-un pachet */
                count = bytes — cptr->byte->count;
                m=0;                                        /* ultimul pachet al acestui mesaj */
            }
            for (i=0; i<count; i++) data[i]= bufptr[cptr->byte_count+1];
            to_net(cid, 0, m,DATA_PKT, data, count);                    /* trimite un pachet */
            cptr->byte_count=cptr->byte_count + count;/* incrementează nr. octeţilor trimişi
            */
        } while (cptr->byte_count < bytes); /* ciclează până când întregul mesaj este trimis */
        cptr->credits--;                                    /* fiecare mesaj foloseşte un credit */
        cptr->state = ESTABLISHED;
```

```
        return(OK);
    } else { cptr->state = ESTABLISHED;
        return(ERR_CLOSED); /* întoarce insucces: celălalt capăt vrea să se deconecteze */ }
}

int receive (int cid, unsigned char bufptr[], int *bytes)
{
    /* Utilizatorul este gata să primească un mesaj */
    struct conn *cptr = &conn[cid];

    if (cptr->clr_req_received == 0) { /* Conexiunea încă stabilită; încearcă să recepţioneze */
        cptr->state = RECEIVING;
        cptr->user_buf_addr = bufptr; cptr->byte_count = 0; data[0] = CRED;
        data[1] = 1;
        to_net(cid, 1, 0, CREDIT, data, 2);                          /* trimite CREDIT */
        sleep();                                        /* se blochează în aşteptarea datelor */
        *bytes = cptr->byte_count;
    }
    cptr->state = ESTABLISHED; return(cptr->clr_req_received ? ERR_CLOSED : OK);
}

int disconnect (int cid)
{   /* Utilizatorul vrea să se deconecteze */
    struct conn *cptr = &conn[cid];

    if (cptr -> clr_req_received) {                  /* cealaltă parte a iniţiat deconectarea */
        cptr->state = IDLE;                                  /* conexiunea este eliberată */
        to_net(cid, 0, 0, CLEAR_CONF, data, 0);
    } else {                                                    /* se iniţiază terminarea */
        cptr->state = DISCONNECT;                    /* conexiunea nu este eliberată până când
                                                          cealaltă parte nu-şi dă acordul */
        to_net(cid, 0, 0, CLEAR_REQ, data, 0);
    }
    return (OK);
}

void packet_arrival (void)
{   /* A sosit un pachet; urmează prelucrarea lui */
    int cid;                                         /* conexiunea pe care a sosit pachetul */

    int count, i q, m;
    pkt_type ptype;                                   /* CALL_REQ, CALL_ACC, CLEAR_REQ,
 CLEAR_CONF, DATA_PKT, CREDIT */ unsigned char data [MAX_MKT_SIZE];  /* date din pachetul
                                                                          care soseşte */
    struct conn *cptr;

    from_net(&cid, &q, &ptype, data, &count);                    /* preia pachetul */
    cptr = &conn[cid];
    switch (ptype) {
        case CALL_REQ:          /* utilizatorul de la distanţă vrea să stabilească o conexiune */
            cptr->local_address = data[0];
            cptr->remote_address = data[ 1 ];
            if (cptr->local_address == listen_address) {
                listen_conn = ci d;
```

```
                cptr->state = ESTABLISHED; wakeup();
            } else {
                cptr->state = QUEUED; cptr->timer = TIMEOUT;
            }
            cptr->clr_req_received = 0; cptr->credits = 0; break;

        case CALL_ACC:              /* utilizatorul de la distanţă a acceptat CALL_REQ trimis */
            cptr->state = ESTABLISHED;
            wakeup();
            break;

        case CLEAR_REQ:
                /* utilizatorul de la distanţă vrea să se deconecteze sau să refuze un apel */
            cptr->clr_req_received = 1;
            if (cptr->state == DISCONN) cptr->state = IDLE;
            if (cptr->state == WAITING II cptr->state == RECEIVING
                            II cptr->state == SENDING) wakeup();
            break;

        case CLEAR_CONF:            /* utilizatorul de la distanţă acceptă deconectarea */
            cptr->state = IDLE;
            break;

        case CREDIT:                        /* utilizatorul de la distanţă aşteaptă date */
            cptr->credits +=data[1];
            if (cptr->state == SENDING) wakeup();
            break;

        case DATA_PKT:                                          /* au fost trimise date */
            for (i=0; i<count; i++)
                cptr->user_buff_addr[cptr->byte_count + i] = data[i ]; cptr->byte_count +=
            count;
            if (m == 0) wakeup();
    }
}

void clock( void)
{               /* la fiecare interval de timp al ceasului se verifică eventualele
                   depăşiri ale timpilor de aşteptare pentru cererile aflate în starea QUEUED */
    int i;
    struct conn *cptr;

    for (i=1 ; i<=MAX_CONN, i++) {
        cptr = &conn[i];
        if (cptr->timer > 0) {                                  /* ceasul funcţiona */
            cptr->timer --;
            if (cptr->timer == 0) {                             /* timpul a expirat */
                cptr->state = IDLE;
                to_net(i, 0, 0, CLEAR_REQ, data, 0);
            }
        }
    }
}
```

Fig. 6-20. Entitatea de transport aleasă ca exemplu.

Existenţa bitului *Q* în antetul pachetului ne permite să evităm timpul suplimentar introdus de antetul protocolului de transport. Mesajele de date obişnuite sunt trimise ca pachete de date cu *Q*=0. Mesajele legate de protocolul de transport, dintre care există numai unul (CREDIT) în exemplul nostru, sunt trimise ca pachete de date cu Q=1. Aceste mesaje de control sunt detectate şi prelucrate de entitatea de transport receptoare.

Structura de date de bază folosită de entitatea de transport este vectorul *conn,* care are câte o înregistrare pentru fiecare conexiune posibilă. În înregistrare sunt menţinute starea conexiunii, incluzând adresa de nivel transport la fiecare capăt, numărul de mesaje trimise şi recepţionate pe conexiune, starea curentă, un indicator (pointer) la tamponul utilizator, numărul de octeţi ai mesajului trimis sau recepţionat, un bit indicând dacă utilizatorul aflat la distanţă a apelat DISCONNECT, un ceas, şi un contor folosit pentru a permite transmiterea mesajelor. Nu toate aceste câmpuri sunt folosite în exemplul nostru simplu, dar o entitate de transport completă ar avea nevoie de toate, poate chiar de mai multe. Fiecare element din *conn* este iniţializat cu starea *IDLE* .

Atunci când un utilizator apelează CONNECT, nivelului reţea i se va cere să trimită un pachet CALL_REQUEST către maşina de la distanţă şi utilizatorul este blocat. Atunci când pachetul CALL_REQUEST ajunge de partea cealaltă, entitatea de transport este întreruptă pentru a executa *packet_arrival,* care verifică dacă utilizatorul local ascultă la adresa specificată. Dacă da, atunci este trimis înapoi un pachet CALL_ACCEPTED şi utilizatorul de la distanţă este trezit; dacă nu, atunci cererea CALL_REQUEST este pusă într-o coadă pentru un interval de timp TIMEOUT. Dacă în acest interval este făcut un apel LISTEN, atunci conexiunea este stabilită; dacă nu, conexiunea este refuzată la sfârşitul intervalului de timp cu un pachet CLEAR_REQUEST ca să nu fie blocată pe timp nedefinit.

Deşi am eliminat antetul pentru protocolul de transport, tot mai trebuie găsită o modalitate pentru a determina cărei conexiuni transport îi aparţine un anumit pachet, deoarece pot exista simultan mai multe conexiuni. Cea mai simplă soluţie este folosirea numărului de circuit virtual de la nivel reţea şi ca număr de conexiune transport. În plus, numărul de circuit virtual poate fi folosit şi ca index în vectorul *conn.* Atunci când un pachet soseşte pe circuitul virtual *k* la nivel reţea, el îi va aparţine conexiunii *k* a cărei stare este păstrată în *conn[k].* La iniţierea unei conexiuni, numărul de conexiune este ales de entitatea de transport care a iniţiat-o. Pentru cererile de conexiune, nivelul reţea alege ca număr de conexiune orice număr de circuit virtual care nu a fost încă folosit.

Pentru a evita gestionarea tampoanelor în interiorul entităţii de transport, este folosit un mecanism de gestiune a fluxului diferit de fereastra glisantă tradiţională. Când un utilizator apelează RECEIVE, este trimis un **mesaj de credit** entităţii de transport de pe maşina care va transmite şi este înregistrat în vectorul *conn.* Atunci când este apelat SEND, entitatea de transport verifică dacă a sosit vreun credit pe conexiunea respectivă. Dacă da, mesajul este trimis (chiar şi în mai multe pachete, dacă este cazul) şi credit este decrementat; dacă nu, atunci entitatea de transport se blochează până la sosirea unui credit. Mecanismul garantează că nici un mesaj nu este trimis decât dacă cealaltă parte a apelat deja RECEIVE. Ca rezultat, ori de

câte ori soseşte un mesaj, există cu siguranţă un tampon disponibil pentru el. Această schemă poate fi uşor generalizată pentru a permite receptorilor să ofere tampoane multiple şi să ceară mai multe mesaje.

Trebuie reţinută simplitatea codului din fig. 6-20. O entitate de transport reală ar verifica în mod normal validitatea tuturor parametrilor furnizaţi de utilizator, ar putea să revină după căderea reţelei, ar putea rezolva coliziunile la apel şi ar susţine un serviciu transport mult mai general, care ar include facilităţi cum sunt întreruperile, datagramele şi versiunile nonblocante ale primitivelor SEND şi RECEIVE.

6.3.3 Exemplul văzut ca un automat finit

Scrierea unei entităţi de transport este o muncă dificilă şi riguroasă, în special pentru protocoalele reale. Pentru a reduce probabilitatea de apariţie a unei erori, adeseori este util să reprezentăm protocolul ca un automat finit.

Am văzut deja că protocolul nostru folosit ca exemplu are şapte stări pentru fiecare conexiune. Este de asemenea posibil să izolăm cele douăsprezece evenimente care pot să apară pentru a schimba starea unei conexiuni. Cinci dintre aceste evenimente sunt cele cinci primitive ale serviciului de transport. Alte şase sunt reprezentate de sosirile celor şase tipuri distincte de pachete. Ultimul este expirarea intervalului de timp stabilit. Fig. 6-21 arată acţiunile principale ale protocolului sub forma unei matrice. Pe coloane sunt prezentate stările, iar pe linii cele 12 evenimente.

Fiecare intrare în matrice (adică în automatul finit) din fig. 6-21 are până la trei câmpuri: un predicat, o acţiune şi o nouă stare. Predicatul indică condiţiile în care acţiunea este executată. De exemplu, pentru intrarea din colţul stânga-sus, dacă este executat un LISTEN şi nu mai există spaţiu în tabele (predicatul P1), atunci LISTEN eşuează şi starea rămâne aceeaşi. Pe de altă parte, dacă un pachet CALL_REQUEST a sosit deja la adresa de nivel transport la care se face LISTEN (predicatul P2), atunci conexiunea este stabilită imediat. O altă posibilitate este ca P2 să fie fals, adică nici un CALL_REQUEST nu a fost primit, caz în care conexiunea rămâne în starea *IDLE* în aşteptarea unui pachet CALL_REQUEST.

Merită să subliniem că alegerea stărilor folosite în matrice nu este în întregime determinată de protocolul însuşi. În acest exemplu, nu există nici o stare *LISTENING,* care ar fi putut urma în mod normal apelului LISTEN. Nu a fost introdusă o stare *LISTENING* deoarece o stare este asociată cu o intrare în tabela de conexiuni şi la un apel LISTEN nu se creează nici o intrare în tabelă. De ce? Pentru că am decis să folosim identificatorii circuitelor virtuale de la nivel reţea ca identificatori pentru conexiune şi, pentru un apel LISTEN, numărul circuitului virtual este în cele din urmă ales de nivelul reţea atunci când soseşte un CALL_REQUEST.

Acţiunile de la A1 la A12 sunt acţiuni importante, precum trimiterea pachetelor sau rearmarea ceasurilor. Nu sunt menţionate toate acţiunile minore, cum ar fi iniţializarea unor câmpuri ale înregistrării ataşate conexiunii. Dacă o acţiune implică trezirea unui proces în aşteptare, atunci acţiunile care urmează trezirii procesului sunt considerate şi ele. De exemplu, dacă un pachet CALL_RE- QUEST soseşte şi există un proces adormit care îl aşteaptă, atunci transmiterea pachetului CALL_ACCEPT este considerată ca făcând parte din acţiunea care urmează recepţiei lui CALL_REQUEST. După ce este efectuată fiecare acţiune, conexiunea ajunge într-o nouă stare, aşa cum apare şi în fig. 6-21.

Stare		Idle	Waiting	Queued	Established	Sending	Receiving	
C D	LIST	P1: ~1/Idle P2:		~/ Estab				
	CONN	P1: ~/Idle P1 :						
	DISCON				P4: A5/Idle P4:			
	SE				P5: A7/Esta			
	RECE IVE				A9/ Receivin			
	– --	P3: A1/Esta						
	Call		~/Estab					

Predicate	Act iuni	
P1: Tabela de	A1: Trimite	A7: Trimite
P2: Cerere de	A2: Aşteaptă	A8: Aşteaptă
aşteptare	A3: Trimite	A9: Trimite
P3: Apel LISTEN în	A4: Porneste	A10: Setează
P4: Pachet	A5: Trimite	Clr req recei
aşteptare	A6: Trimite	A11:
P5: Credit disponibil		A12: Acceptă

Fig. 6-21. Protocolul ales ca exemplu, reprezentat ca un automat finit. Fiecare intrare are un predicat opţional, o acţiune opţională şi o nouă stare. Caracterul tilda indică faptul că nici o acţiune importantă nu este efectuată. Bara deasupra predicatului este reprezentarea pentru predicatul negat. Intrările vide corespund unor secvenţe de evenimente imposibile sau eronate.

Avantajul reprezentării protocolului ca o matrice este întreit. În primul rând, în această formă este mult mai simplu pentru programator să verifice sistematic fiecare combinaţie stare/ eveniment/ acţiune. În implementările reale, unele combinaţii vor fi folosite pentru tratarea erorilor. În fig. 6-21 nu s-a făcut nici o distincţie între situaţiile imposibile şi cele care sunt numai ilegale. De exemplu, dacă o conexiune este în starea *WAITING,* evenimentul DISCONNECT este imposibil deoarece utilizatorul este blocat şi nu poate să facă nici un apel. Pe de altă parte, în starea *SENDING* nu pot sosi date deoarece nu a fost generat nici un credit. Sosirea unui pachet de date va fi o eroare a protocolului.

Al doilea avantaj al reprezentării protocolului ca o matrice este legat de implementarea acestuia. Într-un vector bidimensional elementul *act[i][j]* ar putea fi văzut ca un indicator la procedura care tratează evenimentul i atunci când starea este j. O implementare posibilă este codificarea entităţii de transport ca o buclă în care se aşteaptă la început producerea unui eveniment. După producerea evenimentului, este identificată conexiunea implicată şi este extrasă starea ei. Cunoscând acum starea şi evenimentul, entitatea de transport nu face decât să acceseze vectorul *act* şi să apeleze procedura adecvată. Această abordare ar conduce la o arhitectură mult mai regulată şi mai simetrică decât cea a entităţii de transport implementate de noi.

Al treilea avantaj al abordării folosind un automat finit este legat de descrierea protocolului. În unele standarde, protocoalele sunt specificate ca un automat finit de tipul celui din fig. 6-21. Trecerea de la acest tip de descriere la o entitate de transport funcţională este mult mai uşoară dacă entitatea de transport este şi ea modelată cu ajutorul unui automat finit.

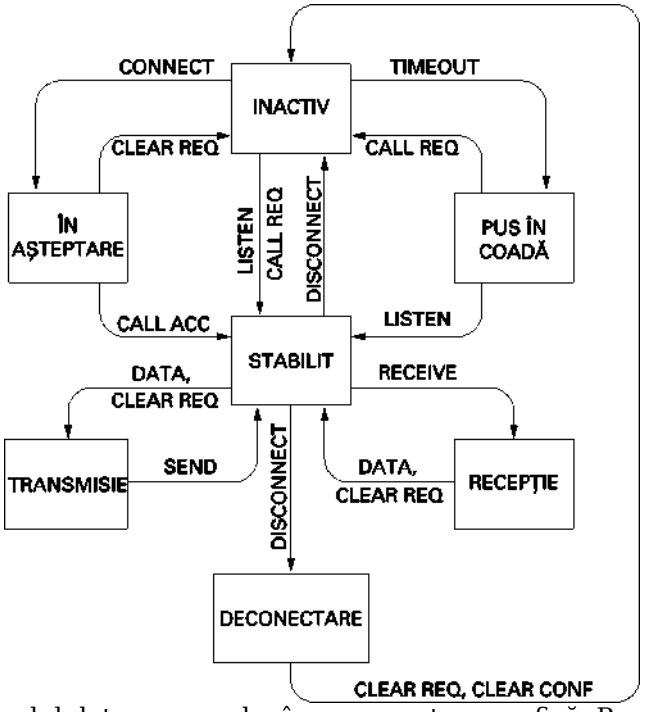


Fig. 6-22. Protocolul dat ca exemplu, în reprezentare grafică. Pentru simplificare, tranziţiile care lasă starea conexiunii nemodificată au fost omise.

Cel mai important dezavantaj este că această abordare poate să fie mai dificil de înţeles decât metoda directă pe care am utilizat-o la început. Totuşi, această problemă poate fi rezolvată, fie şi parţial, desenând automatul finit ca un graf, aşa cum am făcut-o în fig. 6-22.

6.4 PROTOCOALE DE TRANSPORT PRIN IN TERNET: UDP

Internet-ul are două protocoale principale în nivelul de transport: unul neorientat pe conexiune şi unul orientat pe conexiune. În următoarele secţiuni o să le studiem pe ambele. Protocolul neorientat pe conexiune se numeşte UDP. Protocolul orientat pe conexiune se numeşte TCP. O să începem cu UDP-ul deoarece în esenţă este la fel ca IP-ul cu un mic antet adăugat. De asemeni, o să studiem şi două aplicaţii ale UDP-ului.

6.4.1 Introducere în UDP

Setul de protocoale Internet suportă un protocol de transport fără conexiune,

UDP (**User Protocol –** Protocol cu Datagrame Utilizator). UDP oferă aplicaţiilor o modalitate de a trimite datagrame IP încapsulate şi de a le transmite fără a fi nevoie să stabilească o conexiune. UDP este descris în RFC 768.

UDP transmite **segmente** constând într-un antet de 8 octeţi urmat de informaţia utilă . Antetul este prezentat în fig. 6-23. Cele două porturi servesc la identificarea punctelor terminale ale maşinilor sursă şi destinaţie. Când ajunge un pachet UDP, conţinutul său este predat procesului ataşat portului destinaţie. Această ataşare are loc atunci când se foloseşte o simplă procedură de nume sau ceva asemănător, aşa cum am văzut în fig. 6-6 pentru TCP (procesul de legătură este acelaşi pentru UDP). De fapt, valoarea cea mai importantă dată de existenţa UDP-ului faţă de folosirea doar a IP- ului simplu, este aceea a adăugării porturilor sursă şi destinaţie. Fără câmpurile portului, nivelul de transport nu ar şti ce să facă cu pachetul. Cu ajutorul lor, segmentele se livrează corect.

Port sursă	Port destinaţie
Lungime UDP	Sumă de control UDP

Fig. 6-23. Antetul UDP.

Portul sursă este în primul rând necesar atunci când un răspuns trebuie transmis înapoi la sursă. Prin copierea câmpului *port sursă* din segmentul care soseşte în câmpul *port destinaţie* al segmentului care pleacă, procesul ce trimite răspunsul specifică ce proces de pe maşina de trimitere urmează să-l primească.

Câmpul *lungime UDP* include antetul de 8 octeţi şi datele. Câmpul *sumă de control UDP* este opţional şi stocat ca 0 (zero) dacă nu este calculat (o valoare de adevăr 0 rezultată în urma calculelor este memorată ca un şir de biţi 1). Dezactivarea acestuia este o prostie, excepţie făcând cazul în care calitatea informaţiei chiar nu contează (de exemplu, transmisia vocală digitalizată).

Merită probabil menţionate, în mod explicit, unele dintre lucrurile pe care UDP-ul *nu le face*. Nu realizează controlul fluxului, controlul erorii, sau retransmiterea unui segment incorect primit. Toate acestea depind de procesele utilizatorului. Ceea ce face este să ofere protocolului IP o interfaţă cu facilităţi adăugate de demultiplexare a mai multor procese, folosind porturi. Aceasta este tot ceea ce face UDP-ul. Pentru aplicaţiile care trebuie să aibă un control precis asupra fluxului de pachete, controlului erorii sau cronometrarea, UDP-ul furnizează doar ceea ce “a ordonat doctorul”.

Un domeniu unde UDP-ul este în mod special util este acela al situaţiilor client-server. Deseori, clientul trimite o cerinţă scurtă server-ului şi aşteaptă înapoi un răspuns scurt. Dacă se pierde ori cererea ori răspunsul, clientul poate pur şi simplu să încerce din nou după ce a expirat timpul. Nu numai că va fi mai simplu codul, dar sunt necesare şi mai puţine mesaje (câte unul în fiecare direcţie) decât la un protocol care solicită o iniţializare iniţială.

O aplicaţie care foloseşte UDP-ul în acest fel este DNS (Domain Name System, rom: Sistem de rezolvare de nume), pe care îl vom studia în cap. 7. Pe scurt, un

program care trebuie să caute adresele de IP ale unor nume gazdă, de exemplu *www.cs.berkley.edu* , poate trimite un pachet UDP, conţinând numele gazdă, către un server DNS. Serverul răspunde cu un pachet UDP conţinând adresa de IP a gazdei. Nu este necesară nici o iniţializare în avans şi nici o închidere de sesiune. Doar două mesaje traversează reţeaua.

6.4.2 Apel de procedură la distanţă (Remote Procedure Call)

Într-un anume sens, trimiterea unui mesaj către o staţie la distanţă şi primirea înapoi a unui răspuns seamănă mult cu realizarea unei funcţii de apel într-un limbaj de programare. În ambele cazuri se începe cu unul sau mai mulţi parametri şi se primeşte înapoi un rezultat. Această observaţie le-a făcut pe unele persoane să încerce să organizeze interacţiunile cerere-răspuns în reţele pentru a fi puse împreună sub forma apelurilor procedurale. Un astfel de aranjament face aplicaţiile de reţea mai uşor programabile şi mai abordabile. De exemplu, imaginaţi-vă doar procedura numită *get_IP_address(host_name)* care funcţionează prin trimiterea unui pachet UDP către un server DNS şi aşteptarea răspunsului, cronometrând şi încercând încă o dată, dacă răspunsul nu apare suficient de rapid. În acest fel, toate detaliile de reţea pot fi ascunse programatorului.

Efortul cel mai important în acest domeniu a fost depus de către Birell şi Nelson (1984). Rezumând, ce au sugerat Birell şi Nelson a fost să permită programelor să apeleze proceduri localizate pe staţii aflate la distanţă. Când procesul de pe maşina 1 invocă o procedură de pe maşina 2, procesul apelant de pe prima maşină este suspendat şi execuţia procedurii invocate are loc pe cea de-a doua. Informaţia poate fi transportată de la cel care apelează la cel care este apelat în parametri şi se poate întoarce în rezultatul procedurii. Nici un transfer de mesaje nu este vizibil pentru programator. Tehnica este cunoscută sub numele de **RPC** (**Remote Procedure Call** , rom: Apel de procedură la distanţă), şi a devenit baza pentru multe aplicaţii de reţea. În mod tradiţional, procedura care apelează este cunoscută ca fiind clientul şi procedura apelată ca fiind serverul şi vom folosi aceste denumiri şi aici.

Ideea din spatele RPC-ului este aceea de a face un apel de procedură la distanţă să arate pe cât posibil ca unul local. În forma cea mai simplă, pentru apelarea unei proceduri la distanţă, programul client trebuie sa fie legat cu o mică procedură de bibliotecă, numită **client stub** (rom: ciot), care reprezintă procedura server-ului în spaţiul de adresă al clientului. În mod similar, serverul este legat cu o procedură numită **server stub** . Aceste proceduri ascund faptul că apelul de procedură de la client la server nu este local.

Paşii efectivi ai realizării unui RPC sunt prezentaţi în fig. 6-24. Pasul 1 este cel în care clientul apelează stub-ul client. Acest apel este un apel de procedură locală, cu parametrii introduşi în stivă în modul obişnuit. Pasul 2 constă în împachetarea parametrilor de către stub-ul client într-un mesaj şi realizarea unui apel de sistem pentru a trimite mesajul. Împachetarea parametrilor este denumită **marshaling** (rom: împachetare). Pasul 3 constă în faptul că nucleul sistemului de operare trimite un mesaj de la maşina client la maşina server. Pasul 4 constă în trimiterea

de către nucleu a pachetelor care sosesc la stub-ul server. În sfârşit, pasul 5 constă în faptul că stub-ul server apelează procedura server cu parametrii despachetaţi. Răspunsul urmează aceeaşi cale şi în cealaltă direcţie.

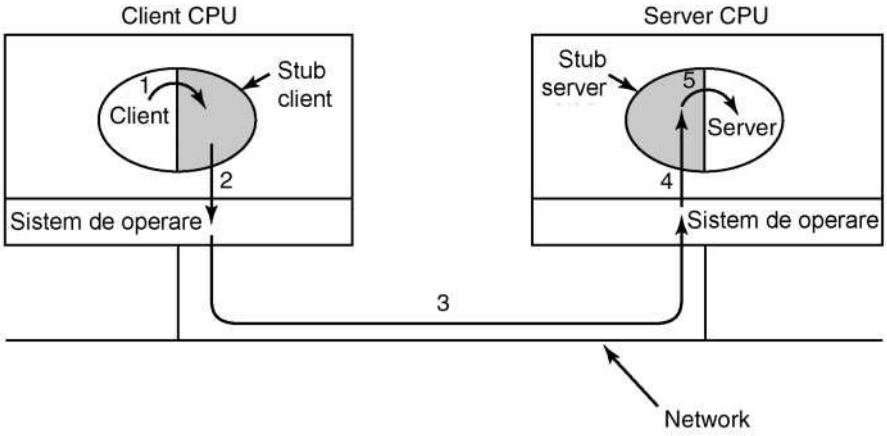


Fig. 6-24. Paşii pentru a crea un apel de procedură la distanţă. Stub-urile sunt haşurate.

Elementul cheie de reţinut aici este acela că procedura client, scrisă de către utilizator, doar face un apel normal de procedură (adică local) către stub-ul client, care are acelaşi nume cu procedura server. Cum procedura client şi stub-ul client se găsesc în acelaşi spaţiu de adresă, parametrii sunt transferaţi în modul obişnuit. În mod asemănător, procedura server este apelată de o procedură din spaţiul său de adresă cu parametrii pe care îi aşteaptă. Pentru procedura server nimic nu este neobişnuit. În felul acesta, în loc ca intrarea/ieşirea să se facă pe socluri, comunicaţia de reţea este realizată imitând o procedură de apelare normală.

În ciuda eleganţei conceptului de RPC, „sunt câţiva şerpi care se ascund prin iarbă”. Unul mare este utilizarea parametrilor pointer. În mod normal, transmiterea unui pointer către procedură nu este o problemă. Procedura apelată poate folosi pointer-ul în acelaşi mod în care poate să o facă şi cel care o apelează, deoarece ambele proceduri se găsesc în acelaşi spaţiu de adrese virtuale. Cu RPC-ul, transmiterea pointer-ilor este imposibilă deoarece clientul şi server-ul se găsesc în spaţii de adresă diferite.

În anumite cazuri, se pot folosi unele trucuri pentru a face posibilă transmiterea pointer-ilor. Să presupunem că primul parametru este un pointer către un întreg, *k.* Stub-ul client poate împacheta variabila *k* şi să o trimită server-ului. Atunci, server stub creează un pointer către *k* şi-l transmite procedurii server, exact aşa cum aceasta se aşteaptă. Când procedura server cedează controlul server stub, acesta din urmă trimite variabila *k* înapoi clientului, unde noul *k* este copiat peste cel vechi, în caz că serverul l-a schimbat. În fapt, secvenţa standard de apelare apel-prin-referinţă a fost înlocuită de copiază-restaurează (eng.: copy-restore). Din păcate, acest truc nu funcţionează întotdeauna, de exemplu dacă un pointer este către un grafic sau altă structură complexă de date. Din acest motiv, trebuie puse anumite restricţii asupra parametrilor procedurilor apelate la distanţă.

O a doua problemă este aceea că în limbajelor mai puţin bazate pe tipuri, cum ar fi C-ul, este perfect legal să scrii o procedură care calculează produsul scalar a doi vectori, fără a specifica dimensiunea vreunuia dintre ei. Fiecare poate fi terminat printr-o valoare specială cunoscută doar de către procedura apelată şi de cea apelantă. În aceste condiţii, în mod cert este imposibil pentru stub-ul client să împacheteze parametrii: nu are nici o modalitate de a determina cât de mult spaţiu ocupă aceştia.

O a treia problemă este aceea că nu întotdeauna este posibilă deducerea tipurilor parametrilor, nici măcar dintr-o specificaţie formală sau din cod în sine. Un exemplu este *printf* care poate avea orice număr de parametri (cel puţin unul), iar parametrii pot fi o combinaţie arbitrară a tipurilor întregi, short, long, caractere, şiruri, numere în virgulă mobilă de diferite lungimi şi alte tipuri. A încerca să invoci *printf* ca procedură cu apel la distanţă ar fi practic imposibil, deoarece C-ul este prea permisiv. Totuşi, o regulă care să spună că RPC-ul poate fi folosit cu condiţia să nu programezi în C (sau C++) nu ar fi prea populară.

O a patra problemă este legată de utilizarea variabilelor globale. În mod normal, procedura de apelare şi cea care este apelată pot comunica folosind variabilele globale, în plus faţă de comunicarea prin parametri. Dacă procedura apelată este mutată acum pe o maşină la distanţă, codul va da erori deoarece variabilele globale nu mai sunt partajate.

Aceste probleme nu sunt menite să sugereze că RPC-ul este lipsit de şanse. De fapt, este larg folosit, dar sunt necesare anumite restricţii pentru a-l face să funcţioneze bine în practică.

Desigur, RPC-ul nu are nevoie să folosească pachete UDP, dar RPC şi UDP se potrivesc bine, şi UDP este uzual folosit pentru RPC. Totuşi, când parametrii sau rezultatele pot să fie mai mari decât pachetul maxim UDP sau atunci când operaţia cerută nu este idempotentă (adică nu poate fi repetată în siguranţă, ca de exemplu atunci când se incrementează un contor), poate fi necesară stabilirea unei conexiuni TCP şi trimiterea cererii prin aceasta, în loc să se folosească UDP-ul.

6.4.3 Protocolul de transport în timp real – Real-Time Transport Protocol

RPC-ul client-server este un domeniu în care UDP este mult folosit. Un alt domeniu este acela al aplicaţiilor multimedia în timp real. În particular, având în vedere că radioul pe internet, telefonia pe Internet, muzica la cerere, videoconferinţele, video la cerere şi alte aplicaţii multimedia au devenit mai răspândite, oamenii au descoperit că fiecare aplicaţie a folosit, mai mult sau mai puţin, acelaşi protocol de transport în timp real. Treptat a devenit clar faptul că un protocol generic de transport în timp real, pentru aplicaţii multimedia, ar fi o idee bună. Aşa a luat naştere **RTP**-ul (**Real-time Transport Protocol**, rom: Protocol de transport în timp real). Este descris în RFC 1889 şi acum este folosit pe scară largă.

Poziţia RTP-ului în stiva de protocoale este oarecum ciudată. S-a hotărât să se pună RTP-ul în spaţiul utilizator şi să se ruleze (în mod normal) peste UDP. El funcţionează după cum urmează. Aplicaţiile multimedia constau în aplicaţii audio,

video, text şi posibil alte fluxuri. Acestea sunt trimise bibliotecii RTP, care se află în spaţiul utilizator împreună cu aplicaţia. Apoi, această bibliotecă multiplexează fluxurile şi le codează în pachete RTP, pe care apoi le trimite printr-un soclu. La celălalt capăt al soclului (în nucleul sistemului de operare), pachete UDP sunt generate şi încapsulate în pachete IP. Dacă computer-ul se găseşte într-o reţea Ethernet, pachetele IP sunt puse apoi în cadre

Ethernet, pentru transmisie. Stiva de protocoale pentru această situaţie este prezentată în fig. 6-25 (a). Încapsularea pachetului este prezentată în fig. 6-25 (b).

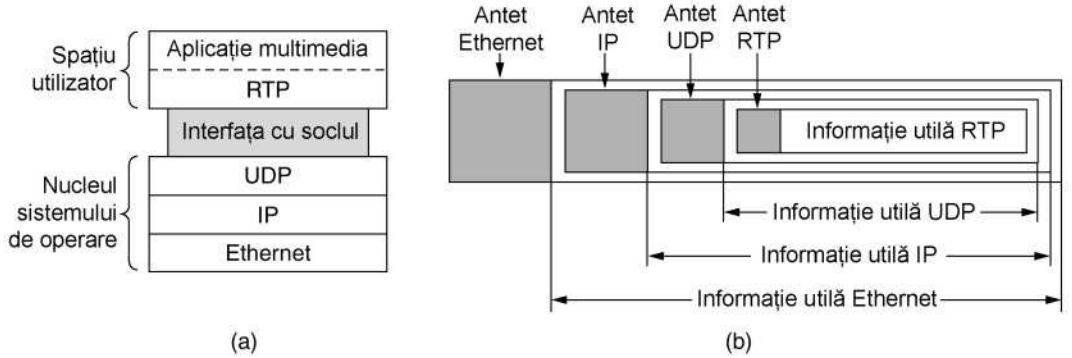


Fig. 6-25. (a)Poziţionarea Protocolului RTP în stiva de protocoale. (b) Încapsularea pachetului.

Ca o consecinţă a acestei proiectări, este cam dificil de spus în ce nivel este RTP-ul. Cum rulează în spaţiul utilizator şi este legat la programul aplicaţie, în mod cert arată ca un protocol de aplicaţie. Pe de altă parte, este un protocol generic independent de aplicaţie, care doar furnizează facilităţi de transport, astfel încât arată totodată ca un protocol de transport. Probabil că cea mai potrivită descriere este aceea că este un protocol de transport care este implementat la nivelul aplicaţie.

Funcţia de bază a RTP-ului este multiplexarea mai multor fluxuri de date în timp real într-un singur flux de pachete UDP. Fluxul UDP poate fi transmis către o singură destinaţie (unicasting) sau către destinaţii multiple (multicasting). Deoarece RTP-ul foloseşte numai UDP normal, pachetele sale nu sunt tratate în mod special de către rutere, decât dacă sunt activate anumite facilităţi de calitate a serviciilor din IP. În particular, nu există garanţii speciale referitoare la livrare, bruiaj etc.

Fiecărui pachet trimis în fluxul RTP i se dă un număr cu unu mai mare decât al predecesorului său. Această numerotare permite destinaţiei să stabilească dacă lipsesc unele pachete. Dacă un pachet lipseşte, cea mai bună decizie ce poate fi luată de către destinaţie este de a aproxima valoarea lipsă prin interpolare. Retransmiterea nu este o opţiune practică având în vedere că pachetul retransmis va ajunge probabil prea târziu pentru a fi util. Ca o consecinţă, RTP-ul nu are control al fluxului, control al erorii, nu are confirmări şi nu are mecanism pentru a cere retransmiterea.

Fiecare informaţie utilă din RTP poate să conţină mostre multiple şi ele pot fi

codate în orice mod doreşte aplicaţia. Pentru a permite compatibilitatea , RTP-ul defineşte mai multe profiluri (de exemplu un singur flux audio) şi pentru fiecare profil pot fi permise multiple formate de codare. De exemplu, un singur flux audio poate fi codat ca mostre de 8 biţi PCM la 8 KHz, codare delta, codare previzibilă, codare GSM, MP3 şi aşa mai departe. RTP-ul furnizează un câmp antet în care sursa poate specifica codarea, dar altfel nu este implicat în modul în care este făcută codarea.

O altă facilitate de care au nevoie multe aplicaţii multimedia este stabilirea amprentei de timp. Aici ideea este de a permite sursei să asocieze o amprentă de timp cu prima mostră din fiecare pachet. Amprentele de timp sunt relative la începutul fluxului, aşa că numai diferenţele dintre acestea sunt semnificative. Valorile absolute nu au nici o semnificaţie. Acest mecanism permite destinaţiei să folosească zone tampon de dimensiuni mici şi să reproducă fiecare eşantion la numărul corect de milisecunde dup ă începutul fluxului, independent de momentul în care ajunge pachetul ce conţine eşantionul. Stabilirea amprentelor de timp nu numai că reduce efectele bruiajului, ci permite de asemenea mai multor fluxuri să se sincronizeze între ele. De exemplu, un program de televiziune digital poate avea un flux video şi două fluxuri audio. Cele două fluxuri audio pot fi pentru emisiuni stereo sau pentru a permite filmelor să fie manipulate cu o coloană sonoră în limba originală şi o coloană sonoră dublată în limba locală, oferind o alegere celui care vede. Fiecare flux provine dintrun dispozitiv fizic diferit, dar dacă sunt stabilite amprentele de timp de către un singur contor, ele pot fi redate sincronizat, chiar dacă fluxurile sunt transmise într-un mod dezordonat.

Antetul RTP este ilustrat în fig. 6-26. Acesta constă din trei cuvinte de 32 biţi şi eventual unele extensii. Primul cuvânt conţine câmpul *Versiune,* care este deja la 2. Să sperăm că această versiune este foarte asemănătoare cu ultima versiune deoarece a mai rămas aici doar un punct de cod (deşi 3 ar putea fi definit ca semnificând faptul că versiunea reală se găseşte într-un cuvânt extins).

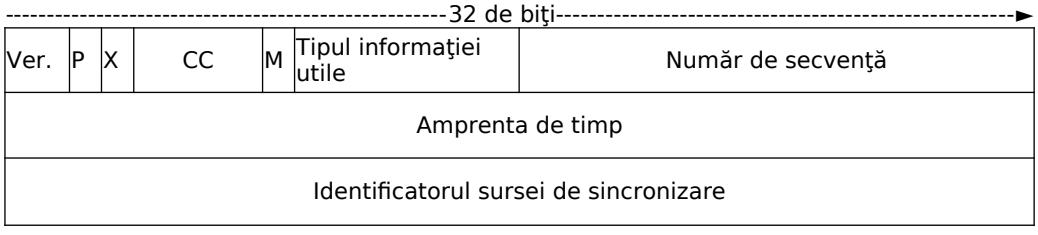


Fig. 6-26. Antetul RTP-ului.

Bitul *P* indică faptul că pachetul a fost extins la un multiplu de 4 octeţi. Ultimul octet extins ne spune câţi octeţi au fost adăugaţi. Bitul *X* indică prezenţa unui antet extins. Formatul şi semnificaţia antetului extins nu sunt definite. Singurul lucru

care este definit este acela că primul cuvânt al extensiei dă lungimea. Aceasta este o cale de scăpare pentru orice cerinţe neprevăzute.

Câmpul *CC* arată câte surse contribuabile sunt prezente, de la 0 la 15 (vezi mai jos). Bitul *M* este un bit de marcare specific aplicaţiei. Poate fi folosit pentru a marca începutul unui cadru video, începutul unui cuvânt într-un canal audio sau altceva ce aplicaţia înţelege. Câmpul *Tip informaţie utilă* indică ce algoritm de codare a fost folosit (de exemplu 8 biţi audio necompresaţi, MP3, etc). Din moment ce fiecare pachet transportă acest câmp, codarea se poate schimba în timpul transmisiei. *Numărul de secvenţă* este doar un contor care este incrementat pe fiecare pachet RTP trimis. Este folosit pentru a detecta pachetele pierdute.

Amprenta de timp este stabilită de către sursa fluxului pentru a se şti când este făcut primul eşantion din pachet. Această valoare poate să ajute la reducerea bruiajului la receptor prin separarea redării de momentul ajungerii pachetului. *Identificatorul sursei de sincronizare* spune cărui flux îi aparţine pachetul. Este metoda utilizată pentru a multiplexa şi demultiplexa mai multe fluxuri de date într-un singur flux de pachete UDP. În sfârşit, dacă există, *identificatorii sursei care contribuie* sunt folosiţi când în studio există mixere. În acest caz, mixerul este sursa de sincronizare şi fluxurile, fiind amestecate, apar aici.

RTP are un protocol înrudit numit **RTCP** (**Real Time Transport Control Protocol**, rom: Protocol de control al transportului în timp real). Acesta se ocupă de răspuns, sincronizare şi de interfaţa cu utilizatorul, dar nu transportă date. Prima funcţie poate fi folosită pentru a oferi surselor reacţie (eng.: feedback) la întârzieri, bruiaj, lăţime de bandă, congestie şi alte proprietăţi ale reţelei. Această informaţie poate fi folosită de către procesul de codare pentru a creşte rata de transfer a datelor (şi să ofere o calitate mai bună) când reţeaua merge bine şi să reducă rata de transfer când apar probleme pe reţea. Prin furnizarea continuă de răspunsuri, algoritmii de codare pot fi în continuu adaptaţi pentru a oferi cea mai bună calitate posibilă în circumstanţele curente. De exemplu, dacă lăţimea de bandă creşte sau scade în timpul transmisiei, codarea poate să se schimbe de la MP3 la PCM pe 8 biţi la codare delta, aşa cum se cere. Câmpul *Tip informaţie utilă* este folosit pentru a spune destinaţiei ce algoritm de codare este folosit pentru pachetul curent, făcând posibilă schimbarea acestuia la cerere.

De asemenea, RTCP-ul se ocupă de sincronizarea între fluxuri. Problema este că fluxuri diferite pot folosi ceasuri diferite, cu granularităţi diferite şi devieri de flux diferite. RTCP poate fi folosit pentru a le menţine sincronizate.

În sfârşit, RTCP oferă un mod pentru a numi diversele surse (de exemplu în text ASCII). Această informaţie poate fi afişată pe ecranul receptorului pentru a indica cine vorbeşte în acel moment.

Mai multe informaţii despre RTP pot fi găsite în (Perkins, 2002).

6.5. PROTOCOALE DE TRANSPORT PRIN

IN TERNET: TCP

UDP-ul este un protocol simplu şi are anumite nişe de utilizare, cum ar fi interacţiunile client-server şi cele multimedia, dar pentru cele mai multe aplicaţii de Internet este necesar un transport de încredere, secvenţial al informaţiei. UDP-ul nu poate oferi acest lucru, deci este nevoie de un alt protocol. Acesta este TCP şi este pionul principal de lucru al Internet-ului. Să-l studiem acum în amănunt.

6.5.1 Introducere în TCP

TCP (**Transport Communication Protocol –** protocol de comunicaţie de nivel transport) a fost proiectat explicit pentru a asigura un flux sigur de octeţi de la un capăt la celălalt al conexiunii într-o inter-reţea nesigură. O inter-reţea diferă de o reţea propriu-zisă prin faptul că diferite părţi ale sale pot diferi substanţial în topologie, lărgime de bandă, întârzieri, dimensiunea pachetelor şi alţi parametri. TCP a fost proiectat să se adapteze în mod dinamic la proprietăţile inter-reţelei şi să fie robust în ceea ce priveşte mai multe tipuri de defecte.

TCP a fost definit în mod oficial în RFC 793. O dată cu trecerea timpului, au fost detectate diverse erori şi inconsistenţe şi au fost modificate cerinţele în anumite subdomenii. Aceste clarificări, precum şi corectarea câtorva erori sunt detaliate în RFC 1122. Extensiile sunt furnizate în RFC 1323.

Fiecare maşină care suportă TCP dispune de o entitate de transport TCP, fie ca proces utilizator, fie ca procedură de bibliotecă, fie ca parte a nucleului. În toate aceste cazuri, ea care gestionează fluxurile TCP şi interfeţele către nivelul IP. O entitate TCP acceptă fluxuri de date utilizator de la procesele locale, le împarte în fragmente care nu depăşesc 64K octeţi (de regulă în fragmente de aproximativ 1460 de octeţi, pentru a încăpea într-un singur cadru Ethernet împreună cu antetele TCP şi IP) şi expediază fiecare fragment ca o datagramă IP separată. Atunci când datagramele IP conţinând informaţie TCP sosesc la o maşină, ele sunt furnizate entităţii TCP, care reconstruieşte fluxul original de octeţi. Pentru simplificare, vom folosi câteodată doar TCP , subînţelegând prin aceasta sau entitatea TCP de transport (o porţiune de program) sau protocolul TCP (un set de reguli). Din context va fi clar care din cele două noţiuni este referită. De exemplu, în „Utilizatorul furnizează date TCP-ului” este clară referirea la entitatea TCP de transport.

Nivelul IP nu oferă nici o garanţie că datagramele vor fi livrate corect, astfel că este sarcina TCP- ului să detecteze eroarea şi să efectueze o retransmisie atunci când situaţia o impune. Datagramele care ajung (totuşi) la destinaţie pot sosi într-o ordine eronată; este, de asemenea, sarcina TCP-ului să le reasambleze în mesaje respectând ordinea corectă (de secvenţă). Pe scurt, TCP-ul trebuie să ofere fiabilitatea pe care cei mai mulţi utilizatori o doresc şi pe care IP-ul nu o oferă.

6.5.2 Modelul serviciului TCP

Serviciul TCP este obţinut prin crearea atât de către emiţător, cât şi de către receptor, a unor puncte finale, numite socluri (sockets), aşa cum s-a discutat în Sec. 6.1.3. Fiecare soclu are un număr de soclu (adresă) format din adresa IP a maşinii gazdă şi un număr de 16 biţi, local gazdei respective, numit **port**. Port este numele TCP pentru un TSAP. Pentru a obţine o conexiune TCP, trebuie stabilită explicit o conexiune între un soclu de pe maşina emiţătoare şi un soclu de pe maşina receptoare. Apelurile de soclu sunt prezentate în fig. 6-5.

Un soclu poate fi folosit la un moment dat pentru mai multe conexiuni. Altfel spus, două sau mai multe conexiuni se pot termina la acelaşi soclu. Conexiunile sunt identificate prin identificatorii soclurilor de la ambele capete, adică *(soclu 1, soclu 2).* Nu este folosit nici un alt număr sau identificator de circuit virtual.

Numerele de port mai mici decât 256 se numesc **porturi general cunoscute** şi sunt rezervate serviciilor standard. De exemplu, orice proces care doreşte să stabilească o conexiune cu o maşină gazdă pentru a transfera un fişier utilizând FTP, se poate conecta la portul 21 al maşinii destinaţie pentru a contacta demonul său FTP. Similar, portul 23 este folosit pentru a stabili o sesiune de lucru la distanţă utilizând TELNET. Lista porturilor general cunoscute se găseşte la *www.iana.org*. Câteva dintre cele foarte cunoscute sunt prezentate în fig. 6-27.

Port	Protocol	Utilitate
21	FTP	Transfer de fi şie re
23	Telnet	Login la distanţă
25	SMTP	E-mail
69	TFTP	Protocol de transfer de fişiere trivial
79	Finger	Căutare de informaţii despre un
80	HTTP	World Wide Web
110	POP-3	Acces prin e-mail la distanţă
119	NNTP	Ştiri USENET

Fig. 6-27. Câteva porturi asignate.

Cu siguranţă ar fi posibil ca, în momentul încărcării, demonul de FTP să se autoataşeze la portul 21, demonul telnet la portul 23 şi tot aşa. Totuşi, dacă s-ar proceda astfel s-ar umple memoria cu demoni inactivi în majoritatea timpului. În schimb, în general se foloseşte un singur demon, numit **inetd** (**Internet daemon**, rom: demon de Internet) în UNIX, care să se autoataşeze la mai multe porturi şi să aştepte prima conexiune care vine. Când acest lucru se întâmplă, inetd creează un nou proces şi execută în el demonul adecvat, lăsând acel demon să se ocupe de cerere. Astfel, demonii, în afară de inetd, sunt activi doar când au de lucru. Inetd află ce porturi să folosească dintr-un fişier de configurare. În consecinţă, administratorul de sistem poate seta sistemul să aibă demoni permanenţi pe cele mai ocupate porturi (de exemplu portul 80) şi inetd pe restul.

Toate conexiunile TCP sunt duplex integral şi punct-la-punct. Duplex integral înseamnă că traficul se poate desfăşura în ambele sensuri în acelaşi timp. Punct-la-punct indică faptul că fiecare conexiune are exact două puncte finale. TCP nu suportă difuzarea parţială sau totală.

O conexiune TCP este un flux de octeţi şi nu un flux de mesaje. Dimensiunile

mesajelor nu se conservă de la un capăt la celălalt. De exemplu, dacă procesul emiţător execută patru scrieri de câte 512 octeţi pe un canal TCP, aceste date pot fi livrate procesului receptor ca patru fragmente (chunks) de 512 octeţi, două fragmente de 1024 octeţi, un singur fragment de 2048 octeţi (vezi fig. 628) sau în orice alt mod. Nu există posibilitatea ca receptorul să determine numărul de unităţi în care a fost scrisă informaţia.

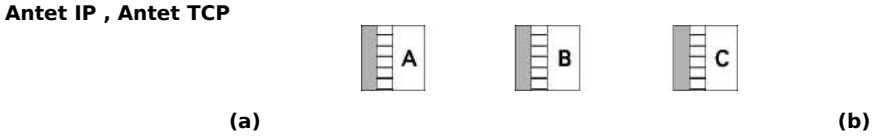


Fig. 6-28. (a) Patru segmente de 512 octeţi au fost trimise ca datagrame IP separate.
(b) Livrarea celor 2048 octeţi către aplicaţie, printr-un singur apel *read.*

În UNIX, aceeaşi proprietate o au şi fişierele. Cititorul unui fişier nu poate spune dacă fişierul a fost scris bloc cu bloc, octet cu octet sau tot dintr-o dată. Ca şi un fişier UNIX, programele TCP nu au nici cea mai vagă idee despre semnificaţia octeţilor şi nici cel mai mic interes pentru a afla acest lucru. Un octet este pur şi simplu un octet.

Atunci când o aplicaţie trimite date către TCP, TCP-ul le poate expedia imediat sau le poate reţine într-un tampon (în scopul colectării unei cantităţi mai mari de informaţie pe care să o expedieze toată odată), după bunul său plac. Cu toate acestea, câteodată, aplicaţia doreşte ca informaţia să fie expediată imediat. De exemplu, să presupunem că un utilizator este conectat la o maşină de la distanţă. După ce a fost terminată o linie de comandă şi s-a tastat Return, este esenţial ca linia să fie imediat expediată către maşina de la distanţă şi să nu fie memorată până la terminarea următoarei linii. Pentru a forţa expedierea, aplicaţia poate folosi indicatorul PUSH, care îi semnalează TCP-ului să nu întârzie procesul de transmisie.

Unele din primele aplicaţii foloseau indicatorul PUSH ca un fel de marcaj pentru a delimita marginile mesajelor. Deşi acest truc funcţionează câteodată, uneori el eşuează datorită faptului că, la recepţie, nu toate implementările TCP-ului transmit aplicaţiei indicatorul PUSH. Mai mult decât atât, dacă mai multe indicatoare PUSH apar înainte ca primul să fi fost transmis (de exemplu, pentru că linia de legătură este ocupată), TCP-ul este liber să colecteze toată informaţia referită de către aceste indicatoare într-o singură datagramă IP, fără să includă nici un separator între diferitele sale părţi.

O ultimă caracteristică a serviciului TCP care merită menţionată aici constă în **informaţia urgentă**. Atunci când un utilizator apasă tasta DEL sau CTRL-C pentru a întrerupe o prelucrare la distanţă, aflată deja în execuţie, aplicaţia emiţător plasează o informaţie de control în fluxul de date şi o furnizează TCP-ului împreună cu indicatorul URGENT. Acest eveniment impune TCP-ului întreruperea acumulării de informaţie şi transmisia imediată a întregii informaţii disponibile deja pentru conexiunea respectivă.

Atunci când informaţia urgentă este recepţionată la destinaţie, aplicaţia receptoare este întreruptă (de ex. prin emisia unui semnal, în terminologie UNIX), astfel încât, eliberată de orice altă activitate, aplicaţia să poată citi fluxul de date şi să poată regăsi informaţia urgentă. Sfârşitul informaţiei urgente este marcat, astfel încât aplicaţia să ştie când se termină informaţia. Începutul informaţiei urgente nu este marcat. Este sarcina aplicaţiei să determine acest început. Această schemă furnizează de fapt un rudiment de mecanism de semnalizare, orice alte detalii fiind lăsate la latitudinea aplicaţiei.

6.5.3 Protocolul TCP

În această secţiune vom prezenta un punct de vedere general asupra protocolului TCP, pentru a ne concentra apoi, în secţiunea care îi urmează, asupra antetului protocolului, câmp cu câmp.

O caracteristică importantă a TCP, care domină structura protocolului, este aceea că fiecare octet al unei conexiuni TCP are propriul său număr de secvenţă, reprezentat pe 32 biţi. Când a luat fiinţă Internetul, liniile dintre rutere erau în cel mai bun caz linii închiriate de 56 Kbps, deci unei gazde funcţionând la viteză maximă îi lua mai mult de o săptămână să utilizeze toate numerele de secvenţă. La vitezele reţelelor moderne, numerele de secvenţă pot fi consumate intr-un ritm alarmant, după cum vom vedea mai târziu. Numerele de secvenţă sunt utilizate atât pentru confirmări cât şi pentru mecanismul de secvenţiere, acesta din urmă utilizând câmpuri separate de 32 de biţi din antet.

Entităţile TCP de transmisie şi de recepţie interschimbă informaţie sub formă de segmente. Un **segment TCP** constă dintr-un antet de exact 20 de octeţi (plus o parte opţională) urmat de zero sau mai mulţi octeţi de date. Programul TCP este cel care decide cât de mari trebuie să fie aceste segmente. El poate acumula informaţie provenită din mai multe scrieri într-un singur segment sau poate fragmenta informaţia provenind dintr-o singură scriere în mai multe segmente. Există două limite care restricţionează dimensiunea unui segment. În primul rând, fiecare segment, inclusiv antetul TCP, trebuie să încapă în cei 65.535 de octeţi de informaţie utilă IP. În al doilea rând, fiecare reţea are o **unitate maximă de transfer** sau **MTU (Maximum Transfer Unit)**, deci fiecare segment trebuie să încapă în acest MTU. În realitate, MTU este în general de 1500 octeţi (dimensiunea informaţiei utile din Ethernet), definind astfel o limită superioară a dimensiunii unui segment.

Protocolul de bază utilizat de către entităţile TCP este protocolul cu fereastră glisantă. Atunci când un emiţător transmite un segment, el porneşte un cronometru. Atunci când un segment ajunge la destinaţie, entitatea TCP receptoare trimite înapoi un segment (cu informaţie utilă, dacă aceasta există sau fără, în caz contrar) care conţine totodată şi numărul de secvenţă următor pe care aceasta se aşteaptă să-l recepţioneze. Dacă cronometrul emiţătorului depăşeşte o anumită valoare înaintea primirii confirmării, emiţătorul retransmite segmentul neconfirmat.

Deşi acest protocol pare simplu, pot apărea multe situaţii particulare pe care le vom prezenta mai jos. Segmentele pot ajunge într-o ordine arbitrară, deci octeţii

3072-4095 pot fi recepţionaţi, dar nu pot fi confirmaţi datorită absenţei octeţilor 2048-3071. Segmentele pot de asemenea întârzia pe drum un interval de timp suficient de mare pentru ca emiţătorul să detecteze o depăşire a cronometrului şi să le retransmită. Retransmisiile pot include porţiuni de mesaj fragmentate altfel decât în transmisia iniţială, ceea ce impune o tratare atentă, astfel încât să se ţină evidenţa octeţilor primiţi corect. Totuşi, deoarece fiecare octet din flux are un deplasament unic faţă de începutul mesajului, acest lucru se poate realiza.

TCP trebuie să fie pregătit să facă faţă unor astfel de situaţii şi să le rezolve într-o manieră eficientă. Un efort considerabil a fost dedicat optimizării performanţelor fluxurilor TCP, ţinându-se cont inclusiv de probleme legate de reţea. În continuare vor fi prezentaţi un număr de algoritmi utilizaţi de numeroase implementări TCP.

6.5.4 Antetul segmentului TCP

În fig. 6-29 este prezentată structura unui segment TCP. Fiecare segment începe cu un antet format dintr-o structură fixă de 20 de octeţi. Antetul fix poate fi urmat de un set de opţiuni asociate antetului. În continuarea opţiunilor, dacă ele există, pot urma până la 65.535 – 20 – 20 = 65.495 de octeţi de date, unde primul 20 reprezintă antetul IP, iar al doilea antetul TCP. Segmente care nu conţin octeţi de date sunt nu numai permise, dar şi utilizate în mod frecvent pentru confirmări şi mesaje de control.

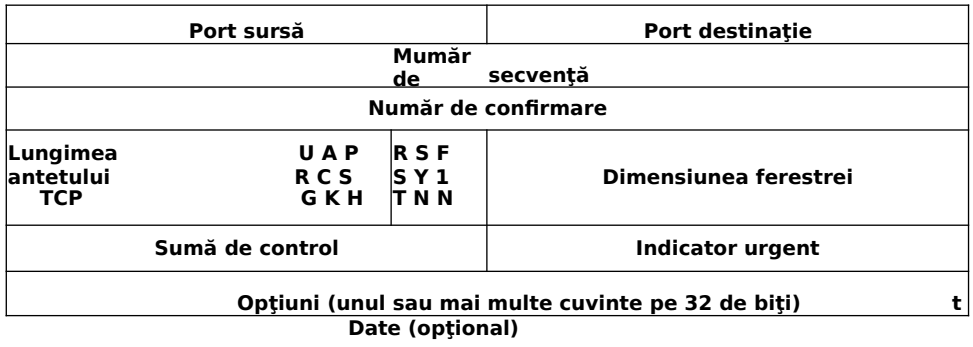


Fig. 6-29. Antetul TCP.

Să disecăm acum structura antetului TCP, câmp cu câmp. Câmpurile *Port sursă* şi *Port destinaţie* identifică punctele finale ale conexiunii. Porturile general cunoscute sunt definite la *www.iana.org*, dar fiecare gazdă le poate aloca pe celelalte după cum doreşte. Un port formează împreună cu adresa IP a maşinii sale un unic punct de capăt (eng.: end point) de 48 de biţi. Conexiunea este identificată de punctele de capăt ale sursei şi destinaţiei.

Câmpurile *Număr de secvenţă* şi *Număr de confirmare* au semnificaţia funcţiilor lor uzuale. Trebuie observat că cel din urmă indică octetul următor aşteptat şi nu ultimul octet recepţionat în mod corect. Ambele câmpuri au lungimea de 32 de biţi,

deoarece într-un flux TCP fiecare bit de informaţie este numerotat.

Lungimea antetului TCP indică numărul de cuvinte de 32 de biţi care sunt conţinute în antetul TCP. Această informaţie este utilă, deoarece câmpul *Opţiuni* este de lungime variabilă, proprietate pe care o transmite astfel şi antetului. Tehnic vorbind, acest câmp indică în realitate începutul datelor din segment, măsurat în cuvinte de 32 de biţi, dar cum acest număr este identic cu lungimea antetului în cuvinte, efectul este acelaşi.

Urmează un câmp de şase biţi care este neutilizat. Faptul că acest câmp a supravieţuit intact mai mult de un sfert de secol este o mărturie despre cât de bine a fost proiectat TCP-ul. Protocoale mai prost concepute ar fi avut nevoie de el pentru a corecta erori ale proiectării iniţiale.

Urmează acum şase indicatori de câte un bit. URG este poziţionat pe 1 dacă Indicatorul Urgent este valid. Indicatorul Urgent este folosit pentru a indica deplasamentul în octeţi faţă de numărul curent de secvenţă la care se găseşte informaţia urgentă. O astfel de facilitate ţine locul mesajelor de întrerupere. Aşa cum am menţionat deja anterior, această facilitate reprezintă esenţa modului în care emiţătorul poate transmite un semnal receptorului fără ca TCP-ul în sine să fie cauza întreruperii.

Bitul *ACK* este poziţionat pe 1 pentru a indica faptul că *Numărul de confirmare* este valid. În cazul în care *ACK* este poziţionat pe 0, segmentul în discuţie nu conţine o confirmare şi câmpul *Număr de confirmare* este ignorat.

Bitul *PSH* indică informaţia FORŢATĂ. Receptorul este rugat respectuos să livreze aplicaţiei informaţia respectivă imediat ce este recepţionată şi să nu o memoreze în aşteptarea umplerii tampoanelor de comunicaţie (lucru care, altminteri, ar fi făcut din raţiuni de eficienţă).

Bitul *RST* este folosit pentru a desfiinţa o conexiune care a devenit inutilizabilă datorită defecţiunii unei maşini sau oricărui alt motiv. El este de asemenea utilizat pentru a refuza un segment invalid sau o încercare de deschidere a unei conexiuni. În general, recepţionarea unui segment având acest bit poziţionat indică o problemă care trebuie tratată în funcţie de context.

Bitul *SYN* este utilizat pentru stabilirea unei conexiuni. Cererea de conexiune conţine *SYN* = 1 şi *ACK* = 0 pentru a indica faptul că acel câmp suplimentar de confirmare nu este utilizat. Răspunsul la o astfel de cerere conţine o confirmare, având deci *SYN* = 1 şi *ACK* = 1. În esenţă, bitul *SYN* este utilizat pentru a indica o CERERE DE CONEXIUNE şi o CONEXIUNE ACCEPTATĂ, bitul *ACK* făcând distincţia între cele două posibilităţi.

Bitul *FIN* este folosit pentru a încheia o conexiune. El indică faptul că emiţătorul nu mai are nici o informaţie de transmis. Cu toate acestea, după închiderea conexiunii, un proces poate recepţiona în continuare date pe o durată nedefinită. Ambele segmente, *SYN* şi *FIN,* conţin numere de secvenţă şi astfel este garantat faptul că ele vor fi prelucrate în ordinea corectă.

În TCP, fluxul de control este tratat prin ferestre glisante de dimensiune variabilă. Câmpul *Fereastră* indică numărul de octeţi care pot fi trimişi, începând de la octetul confirmat. Un câmp *Fereastră* de valoare 0 este perfect legal şi spune că octeţii până la *Număr de confirmare* – 1 inclusiv au fost recepţionaţi, dar receptorul

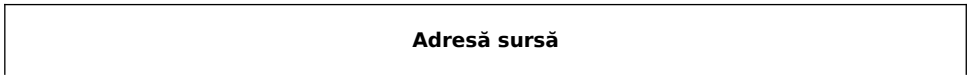
doreşte cu ardoare o pauză, aşa că mulţumeşte frumos, dar pentru moment nu doreşte continuarea transferului. Permisiunea de expediere poate fi acordată ulterior de către receptor prin trimiterea unui segment având acelaşi *Număr de confirmare,* dar un câmp *Fereastră* cu o valoare nenulă.

În protocoalele din cap. 3, confirmările pentru cadrele primite şi permisiunea de a trimite noi cadre erau legate una de alta. Aceasta era o consecinţă a dimensiunii fixe a ferestrei pentru fiecare protocol. În TCP, confirmările şi permisiunea de a trimite noi date sunt total decuplate. De fapt, receptorul poate spune: Am primit octeţii până la al *£*-lea, dar în acest moment nu mai doresc să primesc alţii. Această decuplare (care de fapt reprezintă o fereastră de dimensiune variabilă) oferă mai multă flexibilitate. O vom studia detaliat mai jos.

Este de asemenea prevăzută o *Sumă de control,* în scopul obţinerii unei fiabilităţi extreme. Această sumă de control este calculată pentru antet, informaţie şi pseudo-antetul conceptual prezentat în fig. 6-30. În momentul calculului, *Suma de control* TCP este poziţionată pe zero, iar câmpul de date este completat cu un octet suplimentar nul, dacă lungimea sa este un număr impar. Algoritmul de calcul al sumei de control este simplu, el adunând toate cuvintele de 16 biţi în complement faţă de 1 şi aplicând apoi încă o dată complementul faţă de 1 asupra sumei. În acest mod, atunci când receptorul aplică acelaşi calcul asupra întregului segment, inclusiv asupra *Sumei de control,* rezultatul ar trebui să fie 0.

Pseudo-antetul conţine adresele IP ale maşinii sursă şi destinaţie, de 32 de biţi fiecare, numărul de protocol pentru TCP (6) şi numărul de octeţi al segmentului TCP (incluzând şi antetul). Prin includerea pseudo-antetului în calculul sumei de control TCP se pot detecta pachetele care au fost preluate eronat, dar procedând astfel, este negată însăşi ierarhia protocolului, deoarece adresa IP aparţine nivelului IP şi nu nivelului TCP.

Câmpul *Opţiuni* a fost proiectat pentru a permite adăugarea unor facilităţi suplimentare neacoperite de antetul obişnuit. Cea mai importantă opţiune este aceea care permite fiecărei maşini să specifice încărcarea maximă de informaţie utilă TCP pe care este dispusă să o accepte. Utilizarea segmentelor de dimensiune mare este mai eficientă decât utilizarea segmentelor de dimensiune mică datorită amortizării antetului de 20 de octeţi prin cantitatea mai mare de informaţie utilă. Cu toate acestea, este posibil ca maşini mai puţin performante să nu fie capabile să manevreze segmente foarte mari. În timpul iniţializării conexiunii, fiecare parte anunţă dimensiunea maximă acceptată şi aşteaptă de la partener aceeaşi informaţie. Câştigă cel mai mic dintre cele două numere. Dacă o maşină nu foloseşte această opţiune, cantitatea implicită de informaţie utilă este de 536 octeţi. Toate maşinile din Internet trebuie să accepte segmente de dimensiune 536 + 20 = 556 octeţi. Dimensiunea maximă a segmentului nu trebuie să fie aceeaşi în cele două direcţii.



Adresă destinaţie		
00000000	Protocol = 6	Lungime segment TCP

Fig. 6-30. Pseudo-antetul inclus în suma de control TCP.

O fereastră de 64 K octeţi reprezintă adesea o problemă pentru liniile cu o lărgime de bandă mare şi/sau cu întârzieri mari. Pe o linie T3 (44.736 Mbps) trimiterea unei ferestre întregi de 64 K octeţi durează doar 12 ms. Dacă întârzierea propagării dus-întors este de 50 ms (care este valoarea tipică pentru o linie trans-continentală), emiţătorul va aştepta confirmări – fiind deci inactiv % din timp. Pe o conexiune prin satelit, situaţia este chiar mai rea. O fereastră de dimensiune mare ar permite emiţătorului să continue trimiterea informaţiei, însă o astfel de dimensiune nu poate fi reprezentată în cei 16 biţi ai câmpului Fereastră. În RFC 1323 se propune o opţiune Scală a ferestrei, permiţând emiţătorului şi receptorului să negocieze un factor de scalare a ferestrei. Acest număr permite ambelor părţi să deplaseze câmpul Fereastră cu până la 14 biţi spre stânga, permiţând astfel ferestre de până la 230 octeţi. Această opţiune este suportată în prezent de cele mai multe implementări ale TCP-ului.

O altă opţiune propusă de RFC 1106, şi care este în prezent implementată pe scară largă, constă în utilizarea unei repetări selective în locul unui protocol cu întoarcere de n paşi (eng.: go back n protocol). Dacă receptorul primeşte un segment eronat urmat de un număr mare de segmente corecte, protocolul TCP clasic va constata într-un final o depăşire de timp şi va retrimite toate segmentele neconfirmate, deci şi pe acelea care au fost recepţionate corect (adică se face o întoarcere de n paşi). RFC 1106 introduce NAK-urile pentru a permite receptorului să ceară un anumit segment (sau segmente). După obţinerea acestora, el poate confirma toată informaţia memorată reducând astfel cantitatea de informaţie retransmisă.

6.5.5 Stabilirea conexiunii TCP

În TCP conexiunile sunt stabilite utilizând „înţelegerea în trei paşi”, discutată în Sec. 6.2.2. Pentru a stabili o conexiune, una din părţi – să spunem serverul – aşteaptă în mod pasiv o cerere de conexiune prin execuţia primitivelor LISTEN şi ACCEPT, putând specifica o sursă anume sau nici o sursă în mod particular.

Cealaltă parte – să spunem clientul – execută o primitivă CONNECT, indicând adresa IP şi numărul de port la care doreşte să se conecteze, dimensiunea maximă a segmentului TCP pe care este dispusă să o accepte şi, opţional, o informaţie utilizator (de exemplu o parolă). Primitiva CONNECT trimite un segment TCP având bitul *SYN* poziţionat şi bitul *ACK* nepoziţionat, după care aşteaptă un răspuns.

Atunci când soseşte la destinaţie un segment, entitatea TCP receptoare verifică dacă nu cumva există un proces care a executat LISTEN pe numărul de port specificat în câmpul *Port destinaţie.* În caz contrar, trimite un răspuns cu bitul *RST* poziţionat, pentru a refuza conexiunea.

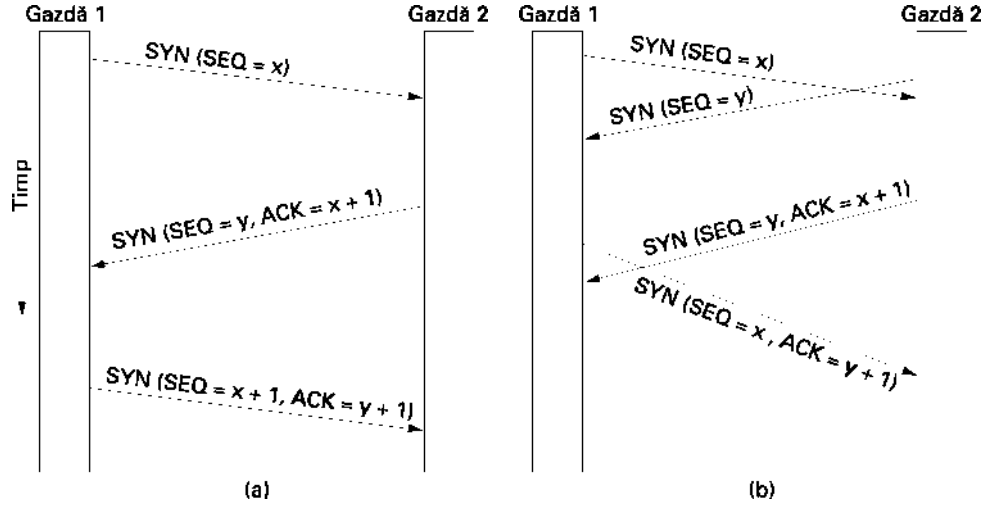

Gazdă 1
Gazdă 2
SYN (SEQ = x)
SYN (SEQ = y, ACK = x + 1)
SYN (SEQ = x + 1, ACK = y + 1)
Timp
(a)
Gazdă 1
Gazdă 2
SYN (SEQ = x)
SYN (SEQ = y)
SYN (SEQ = y, ACK = x + 1)
SYN (SEQ = x, ACK = y + 1)
(b)

Fig. 6-31. (a) Stabilirea unei conexiuni TCP în cazul normal. (b)

Dacă există vreun proces care ascultă la acel port, segmentul TCP recepţionat va fi dirijat către procesul respectiv. Acesta poate accepta sau refuza conexiunea. Dacă o acceptă, trimite înapoi expeditorului un segment de confirmare. În fig. 6-31(a) este reprezentată secvenţa de segmente TCP transferate în caz de funcţionare normală. De notat că un segment *SYN* consumă un octet din spaţiul numerelor de secvenţă, astfel încât confirmarea să poată fi făcută fără ambiguităţi.

Secvenţa de evenimente ilustrată în fig. 6-31(b) reprezintă cazul în care două maşini încearcă simultan să stabilească o conexiune între aceleaşi două porturi. Rezultă că este suficient să fie stabilită o singură conexiune şi nu două, deoarece conexiunile sunt identificate prin punctele lor terminale. Dacă prima iniţializare conduce la crearea unei conexiuni identificată prin *(x, y)* şi acelaşi lucru îl face şi cea de-a doua iniţializare, atunci este construită o singură intrare de tabel, în speţă pentru *(x, y)* .

Numărul iniţial de secvenţă asociat unei conexiuni nu este 0, din motivele discutate anterior. Se utilizează o schemă bazată pe un ceas cu o bătaie la fiecare 4 ps. Pentru mai multă siguranţă , atunci când o maşină se defectează, este posibil ca ea să nu fie reiniţializată în timpul de viaţă maxim al unui pachet, garantându-se astfel că pachetele unei conexiuni anterioare nu se plimbă încă pe undeva prin Internet.

6.5.6 Eliberarea conexiunii TCP

Deşi conexiunile TCP sunt bidirecţionale, pentru a înţelege cum sunt desfiinţate conexiunile, cel mai bine este să ni le imaginăm sub forma unei perechi de legături unidirecţionale. Fiecare legătură unidirecţională este eliberată independent de perechea sa. Pentru eliberarea unei conexiuni, orice partener poate expedia un segment TCP având bitul *FIN* setat, lucru care indică faptul că nici o informaţie nu mai urmează să fie transmisă. Atunci când *FIN*-ul este confirmat, sensul respectiv de comunicare este efectiv oprit pentru noi date. Cu toate acestea, informaţia poate fi transferată în continuare, pentru un timp nedefinit, în celălalt sens. Conexiunea este desfiinţată atunci când ambele direcţii au fost oprite. În mod normal, pentru a elibera o conexiune sunt necesare patru segmente TCP: câte un *FIN* şi un *ACK* pentru fiecare sens. Cu toate acestea, este posibil ca primul *ACK* şi cel de-al doilea *FIN* să fie cuprinse în acelaşi segment reducând astfel numărul total la trei.

La fel ca în conversaţiile telefonice, în care ambele persoane pot spune „la revedere” şi pot închide telefonul simultan, ambele capete ale unei conexiuni TCP pot expedia segmente *FIN* în acelaşi timp. Acestea sunt confirmate ca de obicei, conexiunea fiind astfel eliberată. Nu există de fapt nici o diferenţă esenţială între cazurile în care maşinile eliberează conexiunea secvenţial respectiv simultan.

Pentru a evita problema celor două armate, sunt utilizate cronometre. Dacă un răspuns la un *FIN* nu este recepţionat pe durata a cel mult două cicluri de maxime de viaţă ale unui pachet, emiţătorul *FIN*-ului eliberează conexiunea. Cealaltă parte va observa în final că nimeni nu mai pare să asculte la celălalt capăt al conexiunii, şi va elibera conexiunea în urma expirării unui interval de timp. Această soluţie nu este perfectă, dar având în vedere faptul că o soluţie perfectă este teoretic imposibilă, va trebui să ne mulţumim cu ce avem. În realitate astfel de probleme apar foarte rar.

6.5.7 Modelarea administrării conexiunii TCP

Paşii necesari stabilirii unei conexiuni pot fi reprezentaţi printr-un automat cu stări finite, cele 11 stări ale acestuia fiind prezentate în fig. 6-32. În fiecare stare pot apărea doar anumite evenimente. Atunci când are loc un astfel de eveniment, este îndeplinită o acţiune specifică. Atunci când se produce un eveniment a cărui apariţie nu este legală în starea curentă, este semnalată o eroare.

Stare	Descriere
CLOSED (ÎNCHIS)	Nici o conexiune nu este activă sau în
LISTEN (ASCULTARE)	Serverul aşteaptă recepţionarea unui apel
SYN RCVD (Recepţie SYN)	S-a recepţionat o cerere de conexiune;

SYN SENT (Transmisie SYN)	Aplicaţia a început deschiderea unei
ESTABLISHED (STABILIT)	Starea normală de transfer a datelor
FIN WAIT 1 (Aşteptare FIN 1)	Aplicaţia a anunţat că termină
FIN WAIT 2 (Aşteptare FIN 2)	Partenerul este de acord cu eliberarea
TIMED WAIT (Aşteptare	Se aşteaptă „moartea” tuturor pachetelor
CLOSING (În curs de INCHIDERE)	Ambele părţi încearcă simultan închiderea
CLOSE WAIT (ÎNCHIDE şi	Partenerul a iniţiat eliberarea conexiunii
LAST ACK (CONFIRMARE FINALĂ)	Se aşteaptă „moartea” tuturor pachetelor

Fig. 6-32. Stările utilizate în automatul cu stări finite pentru controlul conexiunii TCP.

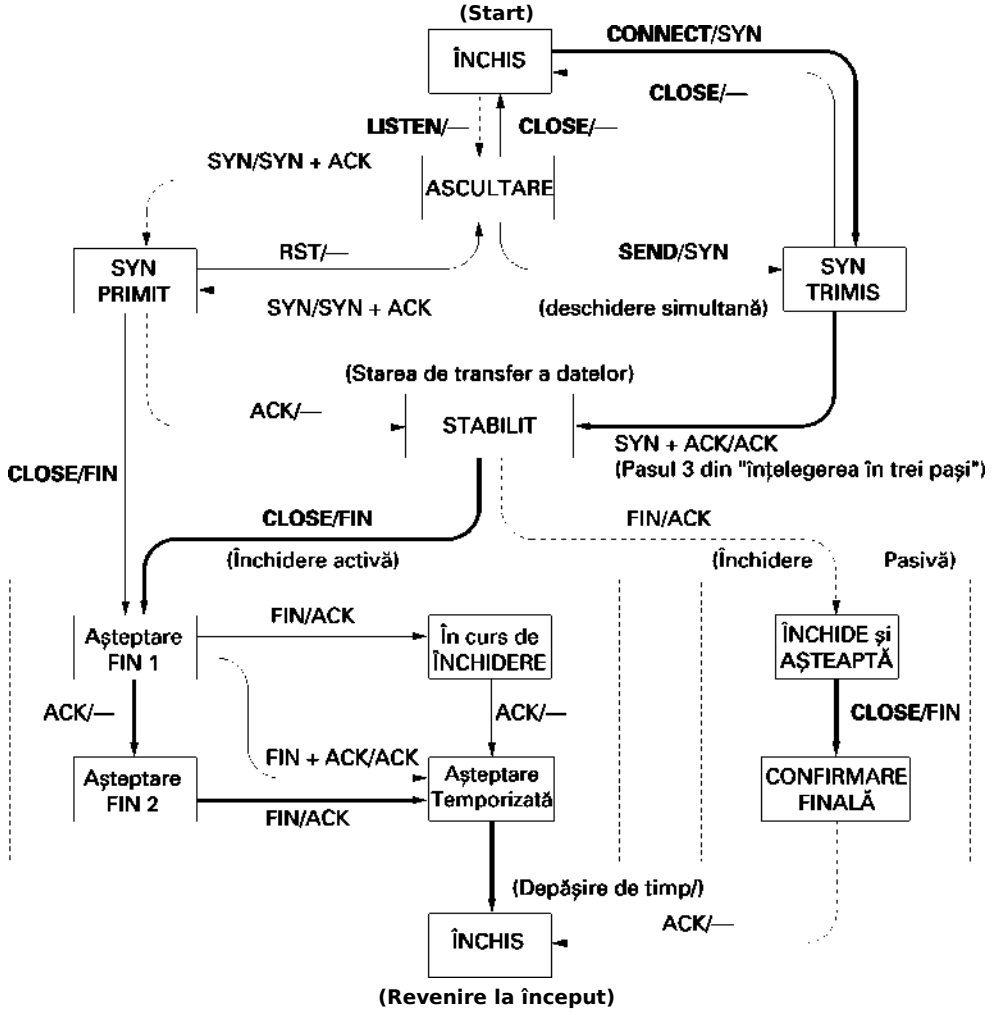


Fig. 6-33. Automatul cu stări finite pentru controlul conexiunii TCP. Linia groasă continuă este calea normală pentru client. Linia groasă întreruptă este calea normală pentru server. Liniile subţiri sunt evenimente neuzuale. Fiecare tranziţie este etichetată de evenimentul care a creat-o şi acţiunea care rezultă din el, separate de slash.

Fiecare conexiune începe în starea *ÎNCHIS.* Această stare este părăsită dacă urmează să se stabilească o conexiune pasivă (LISTEN) sau activă (CONNECT). Dacă partenerul stabileşte o conexiune de tipul opus, starea devine *STABILIT.* Desfiinţarea conexiunii poate fi iniţiată de oricare din parteneri, o dată cu eliberarea conexiunii revenindu-se în starea *ÎNCHIS.*

Automatul cu stări finite este reprezentat în fig. 6-33. Cazul cel mai comun, al unui client conectându-se activ la un server pasiv, este reprezentat prin linii groase – continue pentru client şi întrerupte pentru server. Liniile subţiri reprezintă secvenţe de evenimente mai puţin obişnuite, dar posibile. Fiecare linie din fig. 6-33

este etichetată cu o pereche *eveniment/acţiune.* Evenimentul poate fi unul iniţiat de către utilizator printr-un apel sistem (CONNECT, LISTEN, SEND sau CLOSE), recepţionarea unui segment *(SYN, FIN, ACK* sau *RST)* sau, într-un singur caz, expirarea unui interval de timp egal cu dublul ciclului de viaţă a unui pachet. Acţiunea constă în expedierea unui segment de control *(SYN, FIN* sau *RST)* sau „nici o acţiune", lucru reprezentat prin —. Comentariile sunt incluse între paranteze.

Diagrama poate fi înţeleasă cel mai bine urmărind de la bun început calea urmată de un client (linia groasă continuă) şi apoi calea urmată de un server (linia groasă întreruptă). Atunci când un program aplicaţie de pe maşina client generează o cerere CONNECT, entitatea TCP locală creează o înregistrare de conexiune, o marchează ca fiind în starea *SYN SENT* şi trimite un segment *SYN*. De observat că mai multe conexiuni pot fi deschise (sau în curs de a fi deschise) în acelaşi timp spre folosul mai multor aplicaţii, astfel încât o stare este asociată unei conexiuni şi este înregistrată în înregistrarea asociată acesteia. La recepţia unui *SYN + ACK*, TCP expediază ultima confirmare *(ACK)* din „înţelegerea în trei paşi" şi comută în starea *STABILIT.* Din acest moment, informaţia poate fi atât expediată cât şi recepţionată.

Atunci când se termină o aplicaţie, se apelează primitiva CLOSE care impune entităţii TCP locale expedierea unui segment *FIN* şi aşteptarea *ACK*-ului corespunzător (dreptunghiul figurat cu linie întreruptă şi etichetat „închidere activă"). Atunci când *ACK*-ul este recepţionat, se trece în starea *AŞTEPTARE FIN 2,* unul din sensuri fiind în acest moment închis. Atunci când celălalt sens este la rândul său închis de partenerul de conexiune, se recepţionează un *FIN* care este totodată şi confirmat. În acest moment, ambele sensuri sunt închise, dar TCP-ul aşteaptă un interval de timp egal cu dublul duratei de viaţă a unui pachet, garantând astfel că toate pachetele acestei conexiuni au murit şi că nici o confirmare nu a fost pierdută. Odată ce acest interval de timp expiră, TCP-ul şterge înregistrarea asociată conexiunii.

Să examinăm acum gestiunea conexiunii din punctul de vedere al server-ului. Acesta execută LISTEN şi se „aşează" fiind totodată atent pentru a vedea cine „se ridică în picioare". La recepţiona- rea unui *SYN,* acesta este confirmat şi serverul comută în starea *SYN RCVD.* Atunci când *SYN*-ul server-ului este la rândul său confirmat, „înţelegerea în trei paşi" este completă, serverul comutând în starea *STABILIT.* De acum, transferul informaţiei poate începe.

Atunci când clientul a terminat, execută CLOSE, ceea ce conduce la atenţionarea server-ului prin recepţionarea unui *FIN* (dreptunghiul figurat cu linie întreruptă şi etichetat „închidere pasivă"). Atunci când şi acesta execută un CLOSE, se trimite un *FIN* către client. O dată cu primirea confirmării clientului, serverul desfiinţează conexiunea şi şterge înregistrarea asociată.

6.5.8 Politica TCP de transmisie a datelor

Cum s-a menţionat anterior, administrarea ferestrei în TCP nu este direct legată de confirmări, aşa cum se întâmplă la cele mai multe protocoale de nivel legătură de date. De exemplu, să presupunem că receptorul are un tampon de 4096 octeţi, aşa cum se vede în fig. 6-34. Dacă emiţătorul transmite un segment de 2048 de

octeţi care este recepţionat corect, receptorul va confirma segmentul. Deoarece acum tamponul acestuia din urmă mai are liberi doar 2048 octeţi (până când aplicaţia şterge nişte date din acest tampon), receptorul va anunţa o fereastră de 2048 octeţi începând de la următorul octet aşteptat.

Acum, emiţătorul transmite alţi 2048 octeţi, care sunt confirmaţi, dar fereastra oferită este 0. Emiţătorul trebuie să se oprească până când procesul aplicaţie de pe maşina receptoare a şters nişte date din tampon, moment în care TCP poate oferi o fereastră mai mare.

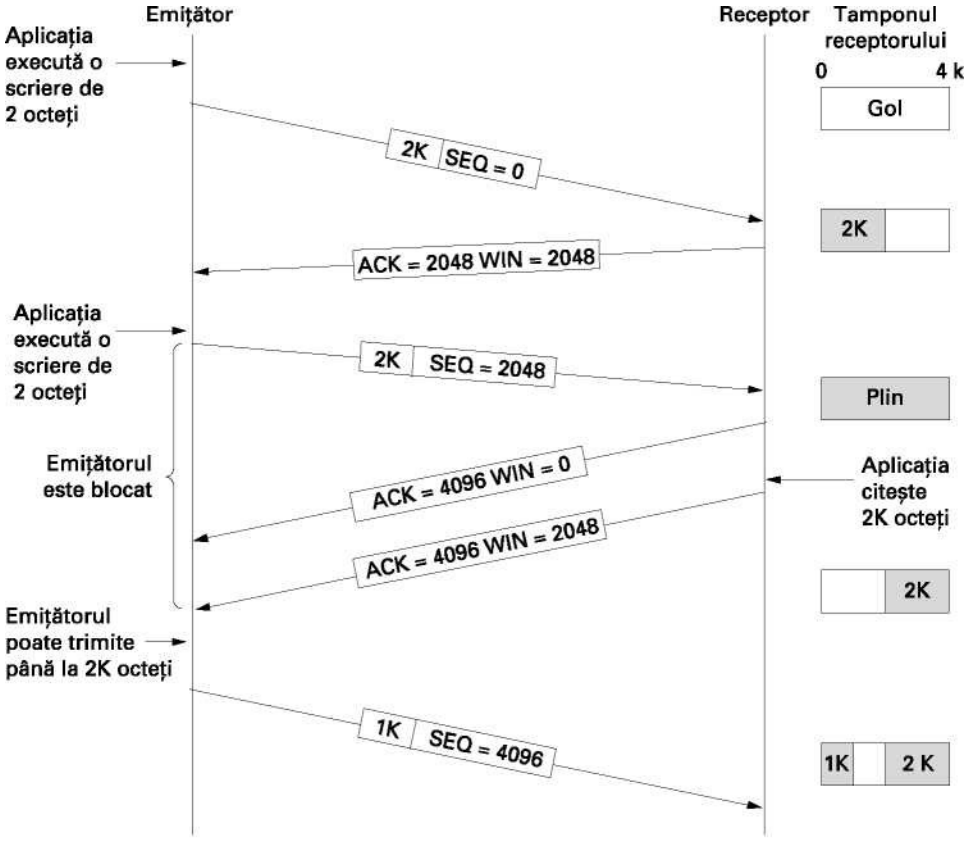


Fig. 6-34. Controlul ferestrei în TCP.

Atunci când fereastra este 0, în mod normal emiţătorul nu poate să transmită segmente, cu două excepţii. În primul rând, informaţia urgentă poate fi trimisă, de exemplu pentru a permite utilizatorului să oprească procesele rulând pe maşina de la distanţă. În al doilea rând, emiţătorul poate trimite un segment de un octet pentru a determina receptorul să renunţe următorul octet aşteptat şi dimensiunea ferestrei. Standardul TCP prevede în mod explicit această opţiune pentru a preveni interblocarea în cazul în care se întâmplă ca anunţarea unei ferestre să fie vreodată pierdută.

Emiţătorii nu transmit în mod obligatoriu date de îndată ce acest lucru este cerut de către aplicaţie. Nici receptorii nu trimit în mod obligatoriu confirmările de îndată ce acest lucru este posibil. De exemplu, în fig. 6-34, atunci când sunt disponibili primii 2K octeţi, TCP, ştiind că dispune de o fereastră de 4K octeţi, va memora informaţia în tampon până când alţi 2K octeţi devin disponibili şi astfel se va putea transmite un segment cu o încărcare utilă de 4K octeţi. Această facilitate poate fi folosită pentru îmbunătăţirea performanţelor.

Să considerăm o conexiune TELNET cu un editor interactiv care reacţionează la fiecare apăsare a tastelor. În cel mai rău caz, atunci când un caracter soseşte la entitatea TCP emiţătoare, TCP creează un segment TCP de 21 octeţi, pe care îl

furnizează IP-ului pentru a fi transmis ca o datagramă IP de 41 octeţi. De partea receptorului, TCP transmite imediat o confirmare de 40 octeţi (20 octeţi antet TCP şi 20 octeţi antet IP). Mai târziu, când editorul a citit caracterul, TCP transmite o actualizare a ferestrei, deplasând fereastra cu un octet la dreapta. Acest pachet este de asemenea de 40 octeţi. În final, când editorul a prelucrat caracterul, transmite ecoul sub forma unui pachet de 41 octeţi. Cu totul, sunt folosiţi 162 octeţi din lărgimea de bandă şi sunt trimise patru segmente pentru orice caracter tipărit. Atunci când lărgimea de bandă este redusă, această metodă de lucru nu este recomandată.

O abordare folosită de multe implementări TCP pentru optimizarea acestei situaţii constă în întârzierea confirmărilor şi actualizărilor de fereastră timp de 500 ms, în speranţa apariţiei unor informaţii la care să se ataşeze pentru o călătorie pe gratis. Presupunând că editorul are un ecou de 50 ms, este necesar acum un singur pachet de 41 octeţi pentru a fi trimis utilizatorului de la distanţă, reducând numărul pachetelor şi utilizarea lărgimii de bandă la jumătate.

Deşi această regulă reduce încărcarea reţelei de către receptor, emiţătorul operează încă ineficient trimiţând pachete de 41 octeţi care conţin un singur octet de date. O modalitate de a reduce această deficienţă este cunoscută ca **algoritmul lui Nagle** (Nagle, 1984). Sugestia lui Nagle este simplă: atunci când emiţătorul dispune de date, în secvenţă, de câte un octet, el va trimite doar primul octet, restul octeţilor fiind memoraţi până la confirmarea primului octet. Apoi vor fi trimise toate caracterele memorate într-un segment TCP şi va continua memorarea până la confirmarea tuturor octeţilor. Dacă utilizatorul tastează repede şi reţeaua este lentă, un număr substanţial de caractere poate fi plasat în fiecare segment, reducând cu mult lărgimea de bandă folosită. În plus, algoritmul permite transmisia unui nou pachet, dacă s-a dispus de suficientă informaţie pentru a umple jumătate dintr-o fereastră sau pentru a completa un segment.

Implementările TCP folosesc pe scară largă algoritmul lui Nagle, dar există situaţii când este mai bine ca el să fie dezactivat. În particular, când o aplicaţie X-Windows rulează prin Internet, deplasările mausului trebuie transmise maşinii de la distanţă. (Sistemul X Window este sistemul de ferestre utilizat pe majoritatea sistemelor UNIX.) Gruparea lor pentru a fi transmise în rafală provoacă o mişcare imprevizibilă a cursorului, lucru care nemulţumeşte profund utilizatorii.

O altă problemă care poate degrada performanţa TCP este **sindromul ferestrei stupide**. (Clark, 1982). Această problemă apare atunci când informaţia este furnizată entităţii TCP emiţătoare în blocuri mari, dar la partea receptoare o aplicaţie interactivă citeşte datele octet cu octet. Pentru a înţelege problema, să analizăm fig. 6-35. Iniţial, tamponul TCP al receptorului este plin şi emiţătorul ştie acest fapt (adică are o fereastră de dimensiune 0). Apoi, aplicaţia interactivă citeşte un caracter de pe canalul TCP. Această acţiune face fericită entitatea TCP receptoare, deci ea va trimite o actualizare de fereastră către emiţător dându-i astfel dreptul de a mai trimite un octet. Îndatorat, emiţătorul trimite un octet. Cu acesta, tamponul este plin şi receptorul confirmă segmentul de 1 octet, dar repoziţionează dimensiunea ferestrei la 0. Acest comportament poate continua la nesfârşit.

Soluţia lui Clark este de a nu permite receptorului să trimită o actualizare de fereastră la fiecare octet. În schimb, el este forţat să aştepte până când spaţiul disponibil are o dimensiune decentă, urmând să-l ofere pe acesta din urmă. Mai precis, receptorul nu ar trebui să trimită o actualizare de fereastră până când nu va putea gestiona minimul dintre dimensiunea maximă oferită atunci când conexiunea a fost stabilită şi jumătate din dimensiunea tamponului său, dacă este liberă.

Mai mult decât atât, emiţătorul poate îmbunătăţi situaţia netrimiţând segmente de dimensiune mică. În schimb, el ar trebui să încerce să aştepte până când acumulează suficient spaţiu în fereastră pentru a trimite un segment întreg sau măcar unul conţinând cel puţin jumătate din dimensiunea tamponului receptorului (pe care trebuie să o estimeze din secvenţa actualizărilor de fereastră recepţionate până acum).

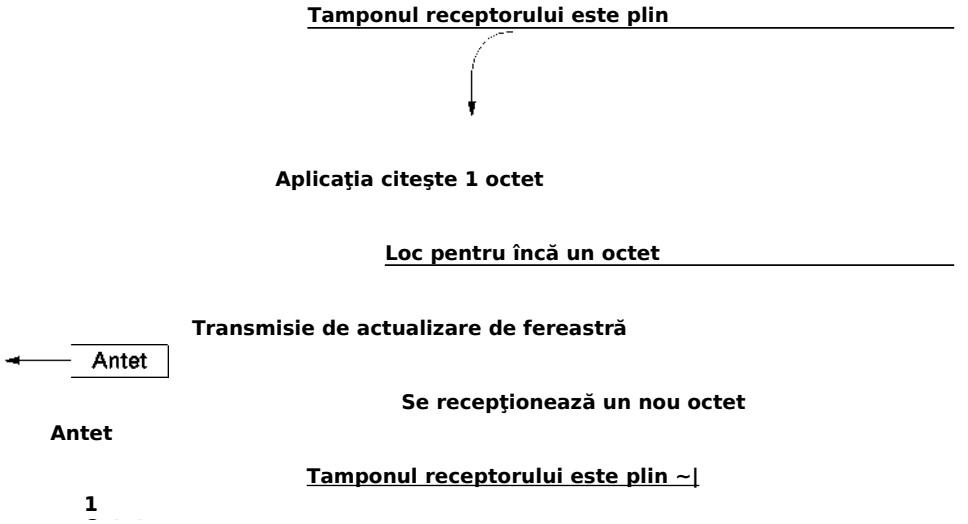


Fig. 6-35. Sindromul ferestrei stupide.

Algoritmul lui Nagle şi soluţia lui Clark pentru sindromul ferestrei stupide sunt complementare. Nagle a încercat să rezolve problema furnizării datelor către TCP octet cu octet, cauzată de aplicaţia emiţătoare. Clark a încercat să rezolve problema extragerii datelor de la TCP octet cu octet, cauzată de către aplicaţia receptoare. Ambele soluţii sunt valide şi pot funcţiona împreună. Scopul este ca emiţătorul să nu trimită segmente mici, iar receptorul să nu ceară astfel de segmente.

Receptorul TCP poate face, pentru îmbunătăţirea performanţelor, mai mult decât simpla actualizare a ferestrei în unităţi mari. Ca şi emiţătorul TCP, el are posibilitatea să memoreze date, astfel încât să poată bloca o cerere de READ a aplicaţiei până când îi poate furniza o cantitate semnificativă de informaţie. Astfel se reduce numărul de apeluri TCP, deci şi supraîncărcarea. Bineînţeles, în acest mod va creşte şi timpul de răspuns, dar pentru aplicaţii care nu sunt interactive, aşa cum este transferul de fişiere, eficienţa poate fi mai importantă decât timpul de răspuns la cereri individuale.

O altă problemă de discutat despre receptor se referă la ce trebuie să facă acesta cu segmentele care nu sosesc în ordine. Ele pot fi reţinute sau eliminate, după cum doreşte receptorul. Bineînţeles, confirmările pot fi trimise numai atunci când toată informaţia până la octetul confirmat a fost recepţionată. Dacă receptorul primeşte segmentele 0, 1, 2, 4, 5, 6 şi 7, el poate confirma totul până la ultimul octet din segmentul 2 inclusiv. Atunci când emiţătorul constată o depăşire de timp, el va retransmite segmentul 3. Dacă receptorul a memorat în tampon segmentele 4 până la 7, odată cu recepţia segmentului 3 el poate confirma toţi octeţii până la sfârşitul segmentului 7.

6.5.9 Controlul congestiei în TCP

Atunci când încărcarea la care este supusă o reţea este mai mare decât poate aceasta să suporte, apare congestia. Internet-ul nu face excepţie. În această secţiune, vom discuta algoritmi care se ocupă cu astfel de congestii şi care au fost dezvoltaţi pe parcursul ultimului sfert de secol. Deşi nivelul reţea

încearcă de asemenea să controleze congestia, cea mai mare parte a muncii este făcută de TCP, şi aceasta deoarece adevărata soluţie a congestiei constă în micşorarea ratei de transfer a informaţiei.

Teoretic, congestia poate fi controlată pe baza unui principiu împrumutat din fizică: legea conservării pachetelor. Ideea de bază este de a nu injecta un nou pachet în reţea până când un pachet mai vechi nu o părăseşte (de exemplu este furnizat receptorului). TCP încearcă să atingă acest scop prin manipularea dinamică a dimensiunii ferestrei.

Primul pas în controlul congestiei este detecţia ei. Mai demult, detecţia congestiei era dificilă. O depăşire de timp datorată pierderii unui pachet putea fi cauzată fie de (1) zgomotul de pe linia de transmisie, fie de (2) eliminarea pachetului de către un ruter congestionat. Diferenţierea celor două cazuri era dificilă.

În zilele noastre, pierderea pachetului din pricina erorilor de transmisie este destul de rară, deoarece cele mai multe din trunchiurile principale de comunicaţie sunt din fibră (deşi reţelele fără fir sunt un subiect separat). În consecinţă, cele mai multe depăşiri ale timpilor de transmisie pe Internet se datorează congestiilor. Toţi algoritmii TCP din Internet presupun că depăşirile de timp sunt cauzate de congestii şi monitorizează aceste depăşiri pentru a detecta problemele.

Înainte de a discuta despre modalitatea în care TCP reacţionează la congestii, să descriem în primul rând modul în care se încearcă prevenirea apariţiei lor. Atunci când se stabileşte o conexiune, trebuie să se aleagă o fereastră de o dimensiune potrivită. Receptorul poate specifica o fereastră bazându-se pe dimensiunea tamponului propriu. Dacă emiţătorul acceptă această dimensiune a ferestrei, nu mai pot apărea probleme datorită depăşirii tamponului la recepţie, dar pot apărea în schimb datorită congestiei interne în reţea.

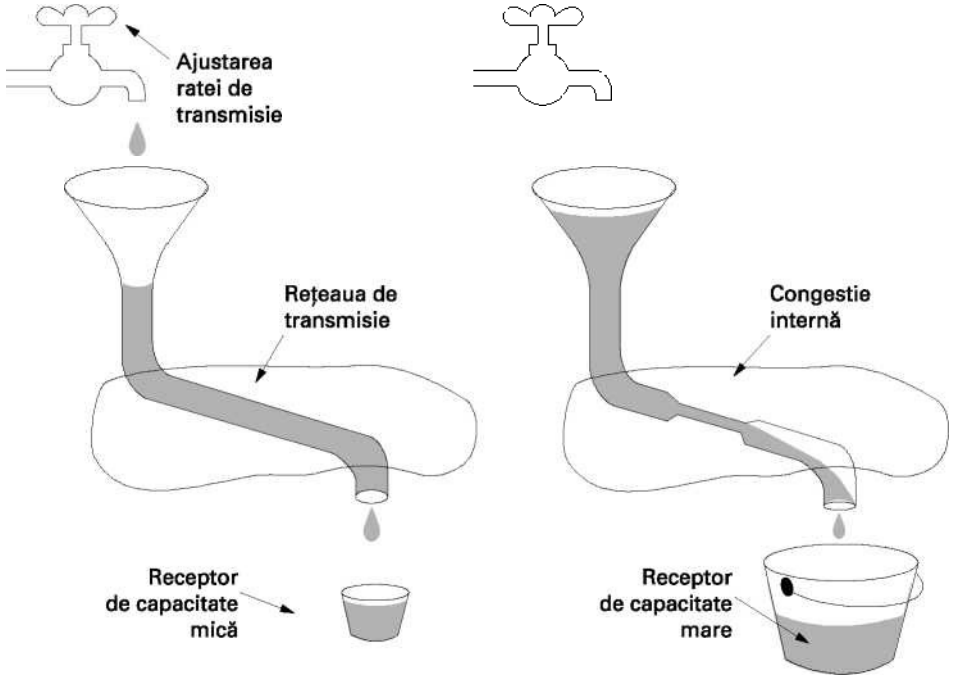


Fig. 6-36. (a) O reţea rapidă alimentând un rezervor de

În fig. 6-36, putem vedea interpretarea hidraulică a acestei probleme. În fig. 6-36(a), observăm o conductă groasă care duce la un receptor de dimensiune mică. Atâta timp cât emiţătorul nu trimite mai multă apă decât poate conţine găleata, apa nu se va pierde. În fig. 6-36(b), factorul de limitare nu mai este capacitatea găleţii, ci capacitatea de transport a reţelei. Dacă vine prea repede prea multă apă, ea se va revărsa şi o anumită cantitate se va pierde (în acest caz prin umplerea pâlniei).

Soluţia din Internet este de a realiza posibilitatea existenţei a două probleme – capacitatea reţelei şi capacitatea receptorului – şi de a le trata pe fiecare separat. Pentru a face acest lucru, fiecare emiţător menţine două ferestre: fereastra acceptată de către receptor şi o a doua fereastră, **fereastra de congestie**. Fiecare reflectă numărul de octeţi care pot fi trimişi de către emiţător. Numărul octeţilor care pot fi trimişi este dat de minimul dintre cele două ferestre. Astfel, fereastra efectivă este minimul dintre ceea ce emiţătorul crede că este „în regulă” şi ceea ce receptorul crede că este „în regulă”. Dacă receptorul spune: „Trimite 8K octeţi”, dar emiţătorul ştie că o rafală de mai mult de 4K octeţi poate aglomera excesiv reţeaua, el va trimite 4K octeţi. Din alt punct de vedere, dacă receptorul spune: „Trimite 8K octeţi” şi emiţătorul ştie că o rafală de 32K octeţi poate străbate fără efort reţeaua, el va trimite toţi cei 8K octeţi ceruţi.

La stabilirea conexiunii, emiţătorul iniţializează fereastra de congestie la dimensiunea celui mai mare segment utilizat de acea conexiune. El trimite apoi un segment de dimensiune maximă. Dacă acest segment este confirmat înaintea expirării timpului, mai adaugă un segment la fereastra de congestie, făcând-o astfel de dimensiunea a două segmente de dimensiune maximă, şi trimite două segmente. O dată cu confirmarea fiecăruia din aceste segmente, fereastra de congestie este redimensionată cu încă un segment de dimensiune maximă. Atunci când fereastra de congestie este de n segmente, dacă toate cele n segmente sunt confirmate în timp util, ea este crescută cu numărul de octeţi corespunzător celor n segmente. De fapt, fiecare rafală confirmată cu succes dublează fereastra de congestie.

Fereastra de congestie creşte în continuare exponenţial până când sau se produce o depăşire de timp, sau se atinge dimensiunea ferestrei receptorului. Ideea este ca dacă rafale de dimensiune, să spunem, 1024, 2048 şi 4096 de octeţi funcţionează fără probleme, dar o rafală de 8192 octeţi duce la o depăşire de timp, fereastra de congestie va fi stabilită la 4096 de octeţi pentru a evita congestia. Atâta timp cât fereastra de congestie rămâne la 4096, nu va fi transmisă nici o rafală mai mare de această valoare, indiferent cât de mult spaţiu de fereastră este oferit de către receptor. Acest algoritm este numit **algoritmul startului lent**, fără a fi însă câtuşi de puţin lent (Jacobson, 1988). Este exponenţial. Toate implementările TCP trebuie să îl suporte.

Să privim acum algoritmul de control al congestiei în cazul Internetului. El utilizează în plus faţă de ferestrele de recepţie şi de congestie un al treilea parametru, numit **prag**, iniţial de 64K. Atunci când apare o depăşire de timp, pragul este poziţionat la jumătate din fereastra curentă de congestie şi fereastra de congestie este repoziţionată la dimensiunea unui segment maxim. Startul lent este utilizat apoi pentru a determina cât poate reţeaua să ducă, atâta doar că acea creştere exponenţială se opreşte odată cu atingerea pragului. De aici înainte

transmisiile reuşite măresc în mod liniar dimensiunea ferestrei de congestie (cu câte un segment maxim pentru fiecare rafală), în locul unei creşteri pentru fiecare segment. De fapt, algoritmul presupune că este acceptabilă înjumătăţirea ferestrei de congestie, din acel punct continuându-şi gradual calea spre dimensiuni mai mari.

Funcţionarea algoritmului de congestie se poate vedea în fig. 6-37. Dimensiunea unui segment maxim este, în acest caz, de 1024 de octeţi. Iniţial fereastra de congestie are 64K octeţi, dar apare o depăşire de timp şi deci pragul este stabilit la 32K octeţi iar fereastra de congestie la 1K octeţi acesta fiind punctul 0 al transmisiei din figură. Fereastra de congestie creşte apoi exponenţial până atinge pragul (32K octeţi). Începând de aici, creşterea este liniară.

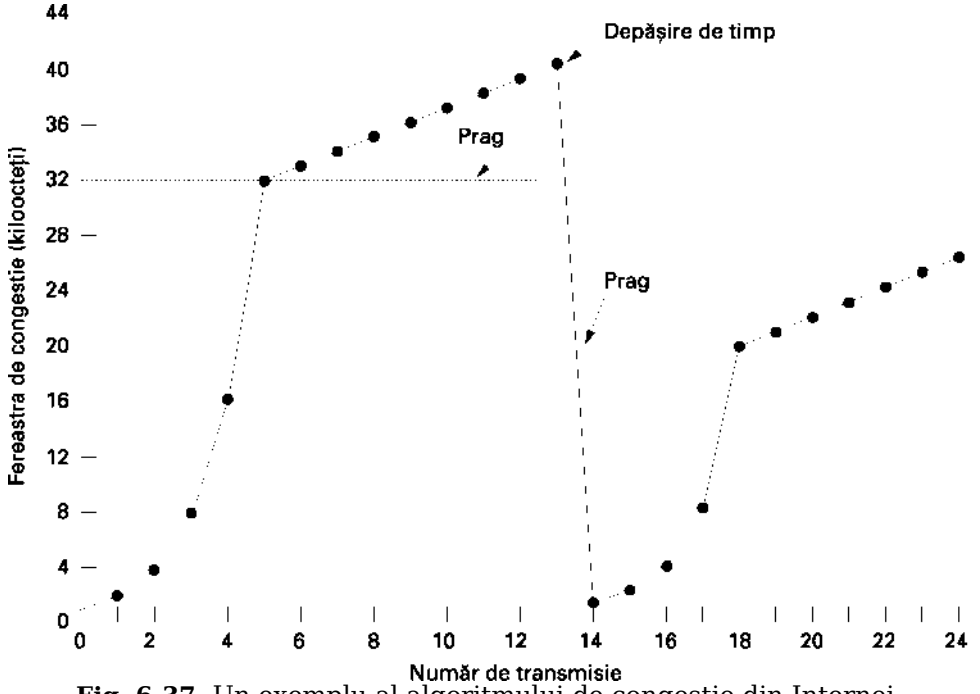


Fig. 6-37. Un exemplu al algoritmului de congestie din Internei.

Transmisia 13 nu are noroc (este de la sine înţeles) şi apare o depăşire de timp. Pragul este stabilit la jumătate din fereastra curentă (acum 40K octeţi, deci jumătate este 20K octeţi) şi startul lent este iniţiat din nou. Atunci când confirmările pentru transmisia 14 încep să sosească, primele patru dublează fiecare fereastra de congestie, dar după aceea creşterea redevine liniară.

Dacă nu mai apar depăşiri de timp, fereastra de congestie va continua să crească până la dimensiunea ferestrei receptorului. În acest punct, creşterea ei va fi oprită şi va rămâne constantă atâta timp cât nu mai apar depăşiri şi fereastra receptorului nu îşi modifică dimensiunea. Ca un alt aspect, dacă un pachet ICMP SOURCE QUENCH soseşte şi este furnizat TCP-ului, acest eveniment este tratat la fel ca o depăşire de timp. O alternativă (şi o abordare mai recentă) este descrisă în RFC 3168.

6.5.10 Administrarea contorului de timp în TCP

TCP utilizează (cel puţin conceptual) mai multe contoare pentru a face ceea ce are de făcut. Cel mai important dintre acestea este **contorul de retransmisie**. Atunci când este trimis un segment, se porneşte un contor de retransmisie. Dacă segmentul este confirmat înainte de expirarea timpului, contorul este oprit. Pe de altă parte, dacă timpul expiră înaintea primirii confirmării, segmentul este retransmis (şi contorul este pornit din nou). Întrebarea care se pune este următoarea: Cât de mare trebuie să fie intervalul de timp până la expirare?

Această problemă este mult mai dificilă la nivelul transport din Internet decât la

nivelul protocoalelor generice de legătură de date prezentate în Cap. 3. În cazul din urmă, întârzierea aşteptată este uşor predictibilă (de exemplu, are o varianţă scăzută), deci contorul poate fi setat să expire chiar imediat după ce era aşteptată confirmarea, aşa cum se arată în fig. 6-38(a). Cum confirmările sunt rareori întârziate în nivelul legătură de date, absenţa unei confirmări în momentul în care aceasta era aşteptată înseamnă de obicei că s-a pierdut fie cadrul, fie confirmarea.

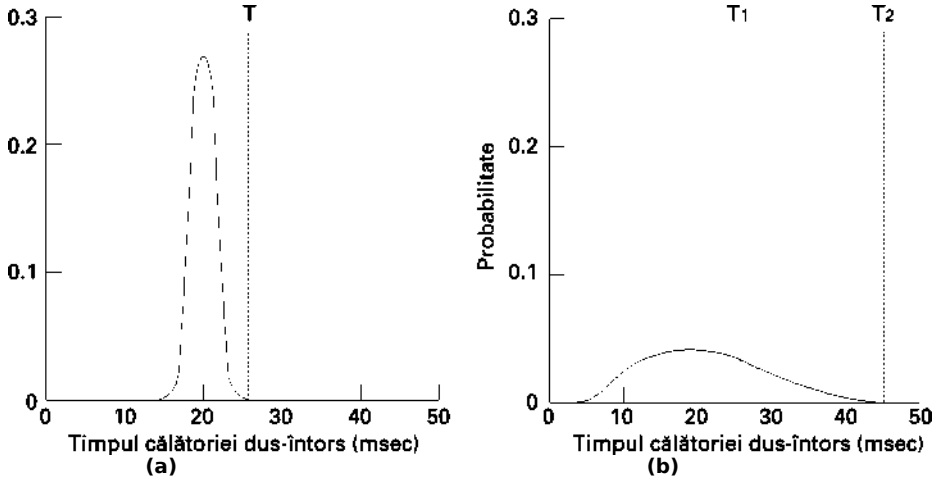


Fig. 6-38. (a) Densitatea de probabilitate a sosirilor în timp a confirmărilor în nivelul legătură de date. (b) Densitatea de probabilitate a sosiri confirmărilor pentru TCP.

TCP trebuie să facă faţă unui mediu radical diferit. Funcţia de densitate a probabilităţii pentru timpul necesar întoarcerii unei confirmări TCP arată mai degrabă ca în fig. 6-38(b) decât ca în fig. 6- 38(a). Este dificilă determinarea timpului în care se realizează circuitul dus-întors până la destinaţie. Chiar şi când acesta este cunoscut, stabilirea intervalului de depăşire este de asemenea dificilă. Dacă intervalul este prea scurt, să spunem T_1 în fig. 6-38(b), vor apărea retransmisii inutile, aglomerând Internet-ul cu pachete fără rost. Dacă este prea lung, (T_2), performanţele vor avea de suferit datorită întârzierii retransmisiei de fiecare dată când se pierde un pachet. Mai mult decât atât, media şi vari- anţa distribuţiei sosirii confirmărilor se pot schimba cu rapiditate pe parcursul a câtorva secunde atunci când apare sau se rezolvă o congestie.

Soluţia este să se utilizeze un algoritm profund dinamic, care ajustează în mod constant intervalul de depăşire bazându-se pe măsurători continue ale performanţei reţelei. Algoritmul utilizat în general de către TCP este datorat lui Jacobson (1988) şi este descris mai jos. Pentru fiecare conexiune, TCP păstrează o variabilă, *RTT,* care este cea mai bună estimare a timpului în care se parcurge circuitul dus-întors către destinaţia în discuţie. Atunci când este trimis un segment, se porneşte un con- tor de timp, atât pentru a vedea cât de mult durează până la primirea confirmării cât şi pentru a iniţia o retransmisie în cazul scurgerii unui interval prea lung. Dacă se primeşte confirmarea înaintea expirării contorului, TCP măsoară cât de mult i-a trebuit confirmării să sosească, fie acest timp *M.* În continuare el actualizează *RTT,*

după formula:

$$RTT = aRTT + (1 - a)M$$

unde a este un factor de netezire care determină ponderea dată vechii valori. Uzual, a=7/8.

Chiar presupunând o valoare bună a lui *RTT*, alegerea unui interval potrivit de retransmisie nu este o sarcină uşoară. În mod normal, TCP utilizează p*RTT*, dar problema constă în alegerea lui p.

În implementările iniţiale, p era întotdeauna 2, dar experienţa a arătat că o valoare constantă este inflexibilă deoarece nu corespunde în cazul creşterii varianţei.

În 1988, Jacobson a propus ca p să fie aproximativ proporţional cu deviaţia standard a funcţiei de densitate a probabilităţii timpului de primire a confirmărilor, astfel încât o varianţă mare implică un P mare şi viceversa. În particular, el a sugerat utilizarea *deviaţiei medii* ca o estimare puţin costisitoare a *deviaţiei standard.* Algoritmul său presupune urmărirea unei alte variabile de netezire, D, deviaţia. La fiecare sosire a unei confirmări, este calculată diferenţa dintre valorile aşteptate şi observate $|A7A - M|$. O valoare netezită a acesteia este păstrată în D, prin formula

$$D = aD + (1 - a)|RTT - M|$$

unde a poate sau nu să aibă aceeaşi valoare ca cea utilizată pentru netezirea lui *RTT.* Deşi *D* nu este chiar deviaţia standard, ea este suficient de bună şi Jacobson a arătat cum poate fi calculată utilizând doar adunări de întregi, scăderi şi deplasări, ceea ce este un mare punct câştigat. Cele mai multe implementări TCP utilizează acum acest algoritm şi stabilesc valoarea intervalului de depăşire la:

$$\text{Depăşire} = RTT + 4 * D$$

Alegerea factorului 4 este într-un fel arbitrară, dar are două avantaje. În primul rând, multiplicarea prin 4 poate fi făcută printr-o singură deplasare. În al doilea rând, minimizează depăşirile de timp şi retransmisiile inutile, datorită faptului că mai puţin de un procent din totalul pachetelor sosesc cu întârzieri mai mari de patru ori deviaţia standard. (De fapt, Jacobson a propus iniţial să se folosească 2, dar experienţa ulterioară a demonstrat că 4 conduce la performanţe mai bune).

O problemă legată de estimarea dinamică a *RTT* se referă la ce trebuie făcut în situaţia în care un segment cauzează o depăşire şi trebuie retransmis. Atunci când confirmarea este primită, nu este clar dacă aceasta se referă la prima transmisie sau la o transmisie următoare. O decizie eronată poate contamina serios estimarea lui *RTT.* Phil Karn a descoperit această problemă cu multă greutate. El este un radio amator entuziast, interesat în transmiterea pachetelor TCP/IP prin operare radio, un mediu recunoscut pentru lipsa de fiabilitate (pe o zi senină, la destinaţie ajung jumătate din pachete). El a făcut o propunere simplă: să nu se actualizeze *RTT* pentru nici un segment care a fost retransmis. În loc de aceasta, timpul de expirare este dublat la fiecare eşec, până când segmentele ajung prima oară la destinaţie. Această corecţie este numită **algoritmul lui Karn**. Cele mai multe din

implementările TCP utilizează acest algoritm.

Contorul de retransmisie nu este singurul utilizat de către TCP. Un al doilea contor este **contorul de persistenţă**. El este proiectat să prevină următoarea situaţie de interblocare. Receptorul trimite o confirmare cu o dimensiune a ferestrei 0, spunându-i emiţătorului să aştepte. Mai târziu, receptorul actualizează fereastra, dar pachetul cu această actualizare este pierdut. Acum, atât emiţătorul cât şi receptorul aşteaptă o reacţie din partea celuilalt. Atunci când contorul de persistenţă expiră, emiţătorul transmite o sondă către receptor. Răspunsul la această investigare furnizează dimensiunea ferestrei. Dacă această dimensiune continuă să fie zero, contorul de persistenţă este repoziţionat şi ciclul se repetă. Dacă este nenul, informaţia poate fi acum transmisă.

Un al treilea contor utilizat de unele implementări este **contorul de menţinere în viaţă**. Când o conexiune a fost inactivă o lungă perioadă de timp, contorul de menţinere în viaţă poate expira pentru a forţa una din părţi să verifice dacă cealaltă parte există încă. Dacă aceasta nu răspunde, conexiunea este închisă. Această facilitate este controversată, deoarece supraîncarcă reţeaua şi poate termina o conexiune altfel sănătoasă datorită unei partiţionări tranzitorii a reţelei.

Ultimul contor folosit la fiecare conexiune TCP este acela utilizat în stările de *AŞTEPTARE CONTORIZATĂ* pe parcursul închiderii conexiunilor. El funcţionează pe durata a două vieţi maxime ale unui pachet, pentru a se asigura că, atunci când o conexiune este închisă, toate pachetele create de aceasta au murit.

6.5.11 TCP şi UDP în conexiune fără fir

Teoretic, protocoalele de transport ar trebui să fie independente de tehnologia nivelului reţea. În particular, TCP-ului ar trebui să nu-i pese dacă IP rulează peste o reţea cablu sau radio. În practică, acest lucru contează, deoarece cele mai multe implementări TCP au fost atent optimizate pe baza unor presupuneri care sunt adevărate pentru reţele cu cabluri, dar care nu mai sunt valabile în cazul reţelelor fără fir. Ignorarea proprietăţilor de transmisie fără fir poate conduce la o implementare TCP corectă din punct de vedere logic, dar cu performanţe incredibil de proaste.

Principala problemă este algoritmul de control al congestiei. Aproape toate implementările TCP din zilele noastre pleacă de la premisa că depăşirile de timp sunt cauzate de congestie şi nu de pierderea pachetelor. În consecinţă, atunci când expiră un contor, TCP încetineşte ritmul şi trimite pachete cu mai puţină vigoare (ex. algoritmul startului lent al lui Jacobson). Ideea din spatele acestei abordări constă în reducerea încărcării reţelei, astfel eliminându-se neplăcerile cauzate de congestie.

Din nefericire, legăturile bazate pe transmisia fără fir nu sunt deloc fiabile. Ele pierd tot timpul pachete. Pentru a controla această pierdere a pachetelor, abordarea corectă este să se retrimită cât mai repede posibil. Încetinirea ritmului nu face decât să înrăutăţească lucrurile. Dacă presupunem că, atunci când emiţătorul transmite 100 de pachete pe secundă, 20% din totalul pachetelor se pierde, productivitatea este de 80 pachete/sec. Dacă emiţătorul încetineşte ritmul la 50

pachete/sec, productivitatea scade la 40 pachete/sec.

Atunci când se pierde un pachet pe o reţea cu cabluri, emiţătorul ar trebui să încetinească ritmul. Atunci când se pierde un pachet pe o reţea fără fir, emiţătorul ar trebui să îl mărească. Dacă emiţătorul nu ştie despre ce tip de reţea este vorba, luarea unei decizii este dificilă.

În mod frecvent, calea de la emiţător la receptor este eterogenă. Primii 1000 km pot să fie într-o reţea cu cabluri, dar ultimul kilometru poate să fie fără fir. Acum, luarea unei decizii în situaţia unei depăşiri de timp este şi mai dificilă, dat fiind că intervine şi locul în care apare problema. O soluţie propusă de Bakne şi Badrinath (1995), **TCP indirect**, constă în spargerea conexiunii TCP în două conexiuni separate, ca în fig. 6-39. Prima conexiune pleacă de la emiţător la staţia de bază. Cea de-a doua leagă staţia de bază de receptor. Această staţie de bază nu face decât să copieze pachetele din cele două conexiuni în ambele direcţii.

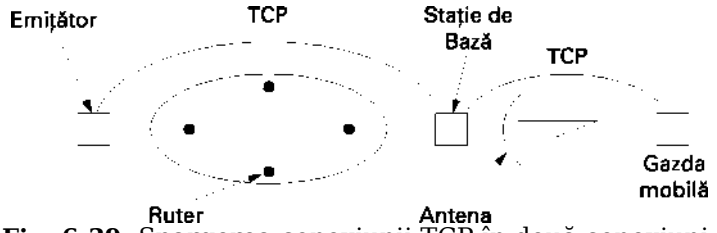


Fig. 6-39. Spargerea conexiunii TCP în două conexiuni.

Avantajul acestei scheme este acela că ambele conexiuni sunt acum omogene. Depăşirile de timp din prima conexiune pot încetini emiţătorul, în timp ce depăşirile de timp din cea de-a doua îl pot accelera. Alţi parametri pot fi de asemenea reglaţi separat în fiecare din cele două conexiuni. Dezavantajul este acela că este negată însăşi semantica TCP. Atâta timp cât fiecare parte a conexiunii este o conexiune TCP în sine, staţia de bază confirmă fiecare segment TCP în mod obişnuit. Doar că acum, recepţia unei confirmări de către emiţător nu mai înseamnă că receptorul a primit segmentul, ci doar că el a fost primit de către staţia de bază.

O soluţie diferită, datorată lui Balakrishnan et. al (1995), nu încalcă semantica TCP. Ea se bazează pe mici modificări făcute în codul nivelului reţea din staţia de bază. Una din modificări constă în adăugarea unui agent de supraveghere care observă şi interceptează pachetele TCP care pleacă spre gazda mobilă precum şi confirmările care se întorc de la acesta. Atunci când observă un segment TCP care pleacă spre gazda mobilă, dar nu observă confirmarea recepţionării acestuia într-un interval de timp dat (relativ scurt), agentul ascuns pur şi simplu retransmite acel segment, fără a mai spune sursei acest lucru. De asemenea, el retransmite şi atunci când observă confirmări duplicate din partea gazdei mobile, lucru care indică invariabil faptul că aceasta a pierdut ceva. Confirmările duplicate sunt eliminate pe loc, pentru a evita ca sursa să le interpreteze ca un semn de congestie.

Cu toate acestea, un dezavantaj al acestei transparenţe este acela că, dacă legătura fără fir pierde multe pachete, sursa poate depăşi limita de timp în aşteptarea unei confirmări şi poate invoca în consecinţă algoritmul de control al congestiei. În cazul TCP-ului indirect, algoritmul de control al congestiei nu va fi niciodată iniţiat dacă nu apare într-adevăr o situaţie de congestie în partea „cabla-

tă" a reţelei.

Algoritmul Balakrishnan oferă de asemenea o soluţie problemei pierderii segmentelor generate de către gazda mobilă. Atunci când staţia de bază constată o pauză în interiorul domeniului numerelor de secvenţă, aceasta generează o cerere pentru o repetare selectivă a octetului lipsă, utilizând o opţiune TCP.

Utilizând aceste corecturi, legătura fără fir devine mai fiabilă în ambele direcţii fără ca sursa să ştie acest lucru şi fără modificarea semanticii TCP.

Deşi UDP-ul nu suferă de aceleaşi probleme ca şi TCP-ul, comunicaţia fără fir induce şi pentru el anumite dificultăţi. Principala problemă este aceea că programele utilizează UDP se aşteaptă ca acesta să fie foarte fiabil. Ele ştiu că nu este furnizată nici o garanţie, dar cu toate acestea se aşteaptă ca el să fie aproape perfect. Într-un mediu fără fir, el va fi însă departe de perfecţiune. Pentru programele care sunt capabile să se refacă după pierderea mesajelor UDP, dar numai cu un cost considerabil, trecerea bruscă de la un mediu în care mesajele puteau fi pierdute mai mult teoretic decât practic la un mediu în care ele sunt pierdute sistematic poate conduce la un dezastru în ceea ce priveşte performanţele.

Comunicaţia fără fir afectează şi alte domenii decât cel al performanţelor. De exemplu, cum poate o gazd ă mobilă să găsească o imprimantă locală la care să se conecteze, în loc să utilizeze propria imprimantă? Oarecum legată de aceasta este şi problema obţinerii paginii WWW pentru celula locală, chiar dacă numele ei nu este cunoscut. De asemenea, proiectanţii paginilor WWW au tendinţa să presupună disponibilă o mare lărgime de bandă. Punerea unei embleme mari pe fiecare pagină poate să devină contraproductivă dacă transmisia paginii printr-o legătură fără fir lentă va dura 10 secunde, şi acest lucru ajunge până la urmă să irite utilizatorii.

Cum reţelele cu comunicaţii fără fir devin tot mai comune, problema rulării TCP-ului pe ele a devenit tot mai acută. Documentaţii suplimentare în acest domeniu se găsesc în (Barakat ş.a., 2000; Ghani şi Dixit, 1999; Huston, 2001; şi Xylomenos ş.a., 2001).

6.5.12 TCP Tranzacţional

Mai devreme în acest capitol am analizat apelul de proceduri la distanţă ca modalitate de a implementa sistemele client-server. Dacă atât cererea, cât şi răspunsul sunt suficient de mici încât să se potrivească în pachete simple şi operaţia este idempotentă, UDP-ul poate fi uşor utilizat. Totuşi, dacă aceste condiţii nu sunt îndeplinite, utilizarea UDP-ului este mai puţin atractivă. De exemplu, dacă răspunsul este unul lung, atunci datagramele trebuie sa fie secvenţiate şi trebuie iniţiat un mecanism pentru a retransmite datagramele pierdute. De fapt, aplicaţiei îi este cerut să reinventeze TCP-ul.

În mod cert, acest lucru nu este atractiv, dar nici utilizarea TCP-ului în sine nu este atractivă.

Problema este eficienţa. Secvenţa normală a pachetelor pentru a face un RPC peste TCP este prezentată în fig. 6-40(a). În cel mai bun caz sunt necesare nouă pachete.

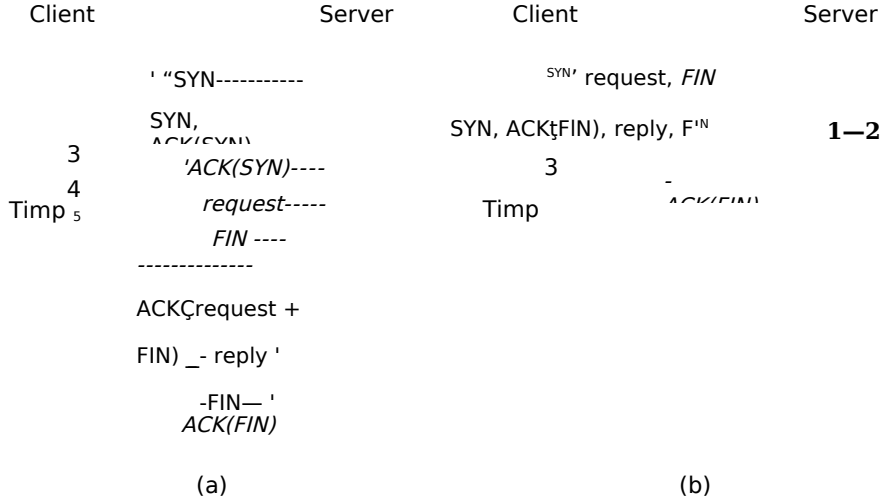


Fig. 6-40. (a) RPC folosind TCP clasic ; (b) RPC folosind T/TCP

Cele nouă pachete sunt după cum urmează:

1. Clientul trimite un pachet SYN pentru a stabili o conexiune.
2. Serverul trimite un pachet ACK pentru a recunoaşte pachetul SYN.
3. Clientul finalizează înţelegerea în trei paşi.
4. Clientul trimite cererea reală.
5. Clientul trimite un pachet FIN pentru a indica dacă s-a terminat trimiterea.
6. Serverul confirmă cererea şi FIN-ul.
7. Serverul trimite răspunsul înapoi clientului.
8. Serverul trimite un pachet FIN pentru a indica că şi acest lucru s-a încheiat.
9. Clientul confirmă FIN-ul server-ului.

A se reţine că acesta este cazul ideal. În cazul cel mai rău, cererea clientului şi FIN-ul sunt confirmate separat, precum sunt răspunsul server-ului şi FIN-ul.

Întrebarea care apare imediat este dacă există vreo posibilitate de a combina eficienţa RPC-ului folosind UDP (doar 2 mesaje) cu fiabilitatea TCP-ului. Răspunsul este: aproape că da. Se poate
realiza cu o variantă experimentală de TCP numită **T/TCP (Transactional TCP**, rom: TCP tranzacţional), care este descrisă în RFC 1379 şi 1644.

Ideea principală este aceea de a modifica secvenţa standard de iniţializare a conexiunii astfel încât să permită, la nivel înalt, transferul de date în timpul iniţializării. Protocolul T/TCP este prezentat în fig. 6-40(b). Primul pachet al clientului conţine bitul SYN, cererea în sine şi FIN-ul. De fapt acesta spune: vreau să stabilesc o conexiune, aici sunt datele şi am terminat

Când serverul primeşte cererea, caută sau calculează răspunsul şi alege modul în care să răspundă. Dacă răspunsul încape într-un pachet, dă răspunsul din fig. 6-40(b), care spune: confirm FIN-ul tău, iată răspunsul, iar eu am terminat. Clientul confirmă apoi FIN-ul server-ului şi protocolul ia sfârşit în (după) trei mesaje.

În orice caz, dacă rezultatul este mai mare de 1 pachet, serverul are de asemenea şi opţiunea de a nu seta bitul FIN, caz în care poate trimite pachete multiple înainte de a-i închide direcţia.

Merită probabil menţionat faptul că T/TCP nu este singura îmbunătăţire propusă pentru TCP. O altă propunere este **SCTP** (**Stream Control Transmission Protocol**, rom: Protocolul de control al transmisiei fluxului). Caracteristicile sale includ păstrarea legăturilor dintre mesaje, modalităţi multiple de livrare (de ex: livrarea neordonată), găzduirea multiplă (destinaţii de rezervă), şi confirmări selective (Stewart and Metz, 2001). În orice caz, oricând cineva propune să schimbe ceva care a funcţionat atât de bine de atâta timp, se duce o luptă aprigă între tabăra “utilizatorii doresc mai multe facilităţi” şi cea “dacă nu e stricat, nu-l repara!”.

6.6 ELEMENTE DE PERFORMANŢĂ

În reţelele de calculatoare sunt foarte importante elementele de performanţă. Atunci când sunt interconectate sute sau mii de calculatoare, au loc adesea interacţiuni complexe cu consecinţe nebănuite. Această complexitate conduce în mod frecvent la performanţe slabe fără ca cineva să ştie de ce. În secţiunile următoare vom examina mai multe elemente legate de performanţa reţelei pentru a identifica tipurile de probleme şi ce poate fi făcut pentru rezolvarea lor.

Din nefericire, înţelegerea performanţei reţelei este mai degrabă o artă decât o ştiinţă. Este prea puţină teorie care stă la bază şi de fapt aceasta nu foloseşte în situaţii practice. Cel mai bun lucru pe care îl putem face este să indicăm reguli rezultate dintr-o experienţă îndelungată şi să prezentăm exemple luate din lumea reală. Am amânat în mod intenţionat această discuţie după studiul nivelului transport din reţelele TCP, pentru a fi capabili să folosim TCP ca exemplu în diverse locuri.

Nivelul transport nu este singurul loc unde apar elemente legate de performanţă. Am văzut unele elemente la nivelul reţea, în capitolul precedent. Cu

toate acestea, nivelul reţea tinde să fie preocupat în mare măsură de rutare şi controlul congestiei. Problemele mai generale, orientate spre sistem ,tind să fie legate de nivelul transport, aşa că acest capitol este locul potrivit pentru a le examina.

În următoarele cinci secţiuni vom examina cinci aspecte de performanţă ale reţelei:

1. Probleme de performanţă.
2. Măsurarea performanţei reţelei.
3. Proiectarea de sistem pentru performanţe mai bune.
4. Prelucrarea rapidă TPDU.
5. Protocoale pentru re ţele viitoare de mare performanţă.

Ca o paranteză, avem nevoie de un nume pentru unităţile schimbate de către entităţile de transport. Termenul TCP de segment este cel mai confuz şi în acest context nu este niciodată utilizat în afara lumii TCP. Termenii uzuali ATM (CS-PDU, SAR-PDU şi CPCS-PDU) sunt specifici doar pentru ATM. Pachetele se referă în mod clar la nivelul reţea şi mesajele aparţin nivelului aplicaţie. Din lipsa unui termen standard, vom reveni la numirea unităţilor schimbate de entităţile de transport TPDU-uri. Când ne referim atât la TPDU cât şi la pachet, vom utiliza pachet ca un termen comun, ca de exemplu în „Procesorul trebuie să fie suficient de rapid pentru a prelucra pachetele în timp real.” Prin aceasta subînţelegem atât pachetul nivelului reţea cât şi TPDU-ul încapsulat în el.

6.6.1 Probleme de performanţă în reţelele de calculatoare

Unele probleme de performanţă, cum este congestia, sunt cauzate de supraîncărcarea temporară a resurselor. Dacă apare subit o creştere de trafic la nivelul unui ruter peste nivelul care poate fi controlat de acesta, se va produce o congestie şi performanţele vor avea de suferit. Congestia a fost studiată în detaliu în capitolul precedent.

Performanţele se degradează de asemenea în cazul unei dezechilibrări structurale a balanţei resurselor. De exemplu, dacă o linie de comunicaţie de un gigabit este ataşată la un terminal PC de performanţă scăzută, procesorul slab nu va putea prelucra suficient de repede pachetele care sosesc, unele din acestea pierzându-se. Aceste pachete vor fi retransmise în cele din urmă, adăugând întârzieri, consumând din lărgimea de bandă şi în general reducând performanţele.

Supraîncărcarea poate fi de asemenea iniţiată în mod sincron. De exemplu, dacă un TPDU conţine un parametru eronat (de exemplu portul pentru care este destinat), în multe cazuri receptorul va înapoia cu multă grijă un anunţ de eroare. Să vedem acum ce s-ar putea întâmpla dacă un TPDU eronat ar fi răspândit către 10000 de maşini: fiecare din ele ar putea întoarce un mesaj de eroare. **Furtuna de difuzare** rezultată ar putea să scoată din funcţiune reţeaua. UDP a suferit de această problemă până când protocolul a fost modificat pentru a împiedica maşinile să răspundă erorilor cauzate de TPDU-urile UDP-ului trimise către adrese de difuzare.

Un al doilea exemplu de supraîncărcare sincronă este dat de efectele unei căderi a energiei electrice. Odată cu revenirea curentului, toate maşinile execută programul ROM de reiniţializare. O secvenţă tipică de reiniţializare poate cere în primul rând unui anume server (DHCP) identitatea precisă a maşinii, apoi poate cere unui server de fişiere o copie a sistemului de operare. Dacă sute de maşini execută acest lucru simultan, serverul va ceda probabil la această încărcare.

Chiar în absenţa unei supraîncărcări sincrone şi chiar atunci când sunt suficiente resurse disponibile, performanţele pot fi slabe datorită unei proaste reglări a sistemului. De exemplu, dacă o maşină are suficientă memorie şi putere de prelucrare, dar nu a fost alocat suficient spaţiu pentru tampoane, vor apărea aglomerări şi se vor pierde TPDU-uri. Similar, dacă algoritmul de planificare nu acordă o prioritate suficient de mare prelucrării TPDU-urilor care sunt recepţionate, unele din ele se vor pierde.

Un alt element de reglare este potrivirea corectă a intervalelor de limită de timp. Atunci când este trimis un TPDU, în mod normal se poziţionează un contor pentru a evita pierderea TPDU-lui. Dacă limita de timp este prea scurtă, se vor produce retransmisii inutile, obturând cablurile. Dacă limita de timp este prea lungă, se vor introduce întârzieri inutile după pierderea unui TPDU. Alţi parametri ce pot fi reglaţi sunt lungimea intervalului de timp după care datele acumulate sunt confirmate, şi numărul retransmisiilor făcute înainte de a se renunţa.

Reţelele gigabit aduc cu ele noi probleme de performanţă. Să considerăm, de exemplu, transmisia unei rafale de date de 64 K octeţi de la San Diego la Boston, pentru a umple tamponul de 64K

octeţi al receptorului. Să presupunem că legătura este de 1 Gbps şi că întârzierea într-un singur sens prin fibra de sticlă este de 20 ms. Iniţial, la momentul $t = 0$, conducta este goală, ca în fig. 6-41 (a). Doar cu 500 ps mai târziu, în fig. 6-41(b), toate TPDU-urile sunt deja plasate pe fibră. Primul TPDU transmis va fi acum undeva în vecinătatea Brawley-ului, încă departe, în California de Sud. Cu toate acestea, transmisia trebuie să se oprească până se primeşte o actualizare de fereastră.

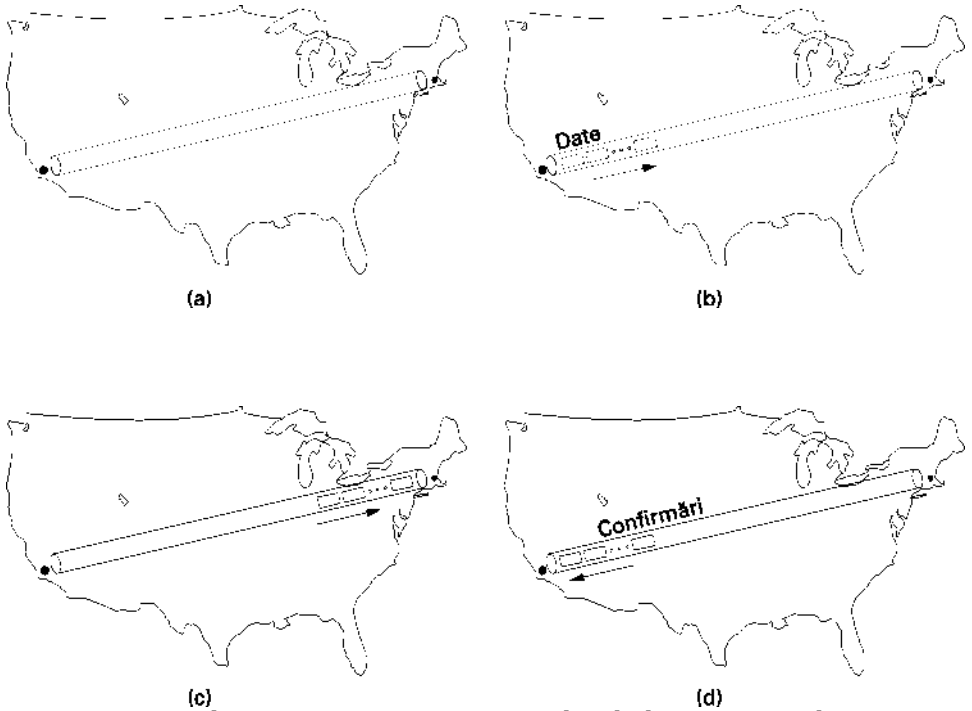


Fig. 6-41. Evoluţia transmisiei unui megabit de la San Diego la Boston. (a) La momentul t=0. (b) După 500 psec. (c) După 20 ms. (d) După 40 ms.

După 20 ms, primul TPDU atinge Boston-ul, ca în fig. 6-41 (c), şi este confirmat. În final, la 40 ms după momentul iniţial, prima confirmare soseşte înapoi la emiţător şi a doua rafală poate să fie transmisă. Cum linia de transmisie a fost utilizată pentru 0.5 ms din cele 40 ms, eficienţa este undeva în jurul a 1.25 procente. Această situaţie este tipică pentru cazul folosirii protocoalelor vechi în linii gigabit.

O mărime utilă de reamintit când se analizează performanţele reţelei este produsul **lărgime de bandă-întârziere**. El este obţinut prin înmulţirea lărgimii de bandă (în biţi pe secundă) cu întârzierea traseului dus-întors (în secunde). Produsul reflectă capacitatea conductei de la emiţător la receptor şi înapoi (în biţi).

De exemplu, în fig. 6-41 produsul lărgime de bandă-întârziere este de 40 milioane de biţi. Cu alte cuvinte, pentru a funcţiona la viteză maximă, emiţătorul poate transmite rafale de 40 milioane de biţi până la recepţionarea primei

confirmări. Umplerea conductei (în ambele sensuri) presupune o cantitate însemnată de biţi. Acesta este motivul pentru care o rafală de jumătate de milion de biţi atinge o eficienţă de 1.25 procente: ea reprezintă doar 1.25 procente din capacitatea conductei.

Concluzia care poate fi trasă aici este că pentru atingerea unor performanţe bune, fereastra receptorului trebuie să fie cel puţin la fel de mare ca şi produsul lărgime de bandă-întârziere, preferabil chiar puţin mai mare, deoarece receptorul poate să nu răspundă instantaneu. Pentru o linie gigabit transcontinentală sunt necesari cel puţin 5 megabiţi.

Dacă eficienţa este dezastruoasă pentru un megabit, imaginaţi-vă cum ar arăta ea pentru o cerere scurtă de câteva sute de octeţi. În cazul în care nu se găseşte o altă utilitate pentru linie în timpul în care primul client este în aşteptarea răspunsului, o linie gigabit nu este cu nimic mai bună ca o linie megabit, numai că este mult mai scumpă.

O altă problemă de performanţă care poate apărea în aplicaţiile critice din punctul de vedere al timpului, precum audio/video, este zgomotul. Un timp mediu de transmisie de valoare mică nu este suficient. Este necesară şi o deviaţie standard mică. Obţinerea atât a unui timp mediu de transmisie mic, cât şi a unei deviaţii standard mici, cere un efort serios de inginerie.

6.6.2 Măsurarea performanţelor reţelei

Atunci când performanţele unei reţele sunt slabe, utilizatorii ei se plâng persoanelor care o administrează, cerând îmbunătăţirea situaţiei. Pentru a creşte performanţele, operatorii trebuie mai întâi să determine cu exactitate ce se întâmplă. Pentru a afla ce se întâmplă în realitate, operatorii trebuie să efectueze măsurători. În această secţiune ne vom concentra asupra măsurării performanţelor reţelei. Discuţia care urmează se bazează pe lucrarea lui Mogul (1993).

Ciclul de bază utilizat pentru îmbunătăţirea performanţelor reţelei conţine următorii paşi:

1. Măsurarea performanţelor şi a parametrilor relevanţi ai reţelei.
2. Încercarea de a înţelege ceea ce se petrece.
3. Modificarea unuia din parametri.

Aceşti paşi se repetă până la atingerea unor performanţe suficient de bune sau până când este clar că şi ultima îmbunătăţire posibilă a fost pusă în aplicare.

Măsurarea poate fi făcută în multe moduri şi în multe locuri (atât fizic cât şi în stiva de protocoale). Cel mai simplu mod de măsurare este iniţializarea unui contor la începutul unei activităţi şi observarea timpului necesar pentru îndeplinirea acelei sarcini. De exemplu, un element cheie în măsurare este aflarea timpului necesar unui TPDU pentru a fi confirmat. Alte măsurători sunt făcute cu contoare care înregistrează frecvenţa de apariţie a unor evenimente (de exemplu, numărul de TPDU-uri pierdute). În final, unii pot fi interesaţi să afle valorile unor mărimi, ca de exemplu numărul de octeţi prelucraţi într-un anume interval de timp.

Măsurarea performanţelor şi a parametrilor reţelei ascunde multe capcane potenţiale. În cele ce urmează, enunţăm doar câteva din acestea. Orice încercare

sistematică de a măsura performanţele reţelei trebuie să le evite.

Dimensiunea testului trebuie să fie suficient de mare

Timpul necesar pentru a trimite un TPDU nu trebuie măsurat o singură dată, ci în mod repetat, să zicem, de un milion de ori, luându-se în considerare media valorilor rezultate. Un test de dimensiune mare va reduce gradul de incertitudine în media şi deviaţia standard a măsurătorii. Această incertitudine poate fi calculată pe baza formulelor statistice obişnuite.

Testele trebuie să fie reprezentative

Ideal ar fi ca întreaga secvenţă a celor un milion de măsurători să fie repetată în diferite momente ale zilei şi ale săptămânii pentru a pune în evidenţă efectul diferenţelor de încărcare a sistemului asupra mărimii măsurate. Măsurătorile de congestie, de exemplu, nu sunt prea utile dacă sunt făcute într-un moment în care nu există nici o congestie. Uneori, rezultatele pot fi iniţial contrare intuiţiei, de exemplu congestii importante la orele 10, 11, sau 1, 2 după amiază, dar nici un fel de congestie la amiază (când toţi utilizatorii au pauză de prânz).

Utilizarea ceasurilor cu granularitate mare trebuie făcută cu atenţie

Ceasurile calculatoarelor funcţionează prin incrementarea la intervale regulate a unui anumit contor. De exemplu, un contor de milisecunde adaugă unu la contor la fiecare 1 ms. Utilizarea unui astfel de contor pentru a măsura un eveniment care durează mai puţin de 1 ms nu este imposibilă, dar necesită o oarecare atenţie. (Desigur, unele calculatoare au ceasuri cu precizie mai bună).

Pentru a măsura intervalul necesar trimiterii unui TPDU, de exemplu, ceasul sistem (să spunem în milisecunde) ar trebui să fie citit în momentul în care se intră în codul de transport şi din nou când se părăseşte acest cod. Dacă timpul real de transmisie TPDU este de 300 psec, diferenţa dintre cele două citiri va fi sau 0, sau 1, ambele valori greşite. Cu toate acestea, dacă măsurătoarea este repetată de un milion de ori şi suma tuturor măsurătorilor este împărţită la un milion, timpul mediu va avea o precizie mai bună de 1 psec.

Nu trebuie să se petreacă ceva neaşteptat în timpul măsurătorilor

Efectuarea măsurătorilor pe un sistem universitar în ziua în care urmează să fie predat vreun proiect important poate conduce la rezultate diferite faţă de cele ce s-ar obţine în ziua imediat următoare. Similar, dacă vreun cercetător se decide să organizeze o videoconferinţă în reţea, în timpul testelor, poate fi obţinut un rezultat alterat. Cel mai bine este ca testele să fie rulate pe un sistem complet inactiv, întreaga sarcină fiind construită în vederea testării. Chiar şi această abordare are propriile capcane. Deşi ne-am aştepta ca nimeni să nu utilizeze reţeaua la ora 3 dimineaţa, acesta poate să fie chiar momentul în care un program de salvare automată începe să copieze conţinutul tuturor discurilor pe bandă. Mai mult decât atât, s-ar putea să existe un trafic important pentru minunatele pagini de Web de pe reţeaua testată, trafic provenit din zone aflate pe alte meridiane orare.

Lucrul cu memoria ascunsă poate distruge măsurătorile

Modalitatea evidentă de a măsura timpul de transfer al fişierelor este de a

deschide un fişier de dimensiune mare, de a-l citi în întregime şi de a-l închide, urmând a vedea cât de mult a durat toată operaţia. Se repetă apoi operaţia de mult mai multe ori, pentru a obţine o medie corectă. Problema este că sistemul poate memora fişierul în memoria ascunsă, astfel încât doar prima măsurătoare să implice traficul în reţea. Restul nu sunt decât accese la memoria ascunsă locală. Rezultatele unei astfel de măsurători sunt, în esenţă, fără nici o valoare (doar dacă nu cumva se doreşte măsurarea performanţelor memoriei ascunse).

De obicei, se poate ocoli memoria ascunsă prin simpla ei supraîncărcare. De exemplu, dacă memoria ascunsă este de 10 megaocteţi, ciclul de test ar putea deschide, citi şi închide două fişiere de 10 megaocteţi la fiecare buclă, în tentativa de a forţa rata de succes în accesul la memoria ascunsă la 0. Cu toate acestea, dacă nu se înţelege cu absolută precizie algoritmul de manipulare a memoriei ascunse, trebuie procedat cu grijă.

Memorarea datelor în tampoane poate avea acelaşi efect. Se cunoaşte un program utilitar popular pentru măsurarea performanţelor TCP/IP care raportează performanţe ale UDP-ului substanţial mai mari decât o permit liniile fizice. Cum se întâmplă acest lucru? Un apel către UDP întoarce în mod normal controlul odată ce mesajul a fost acceptat de către nucleu şi adăugat la coada de transmisie. Dacă este suficient spaţiu în tampon, măsurarea a 1000 de apeluri UDP nu înseamnă neapă

rat că informaţiile au fost transmise. Cea mai mare a informaţiilor poate să se afle încă în nucleu, dar instrumentul de măsurare interpretează că ele au fost toate deja transmise.

Trebuie înţeles ceea ce se măsoară

Atunci când se măsoară timpul necesar pentru a citi un fişier de la distanţă, măsurătorile depind de reţea, de sistemele de operare de la ambele capete – client şi server, de tipul de echipament al plăcii de interfaţă utilizat, de programele care le controlează şi de alţi factori. Procedând cu atenţie, putem determina în ultimă instanţă timpul de transfer al fişierului pentru configuraţia utilizată. Dacă scopul îl reprezintă reglarea acestei configuraţii particulare, atunci măsurătorile sunt în regulă.

Cu toate acestea, dacă, în scopul alegerii unei interfeţe de reţea pentru a fi cumpărată, sunt făcute măsurători similare pe trei sisteme diferite, rezultatele pot fi complet bulversate în cazul în care unul din programele care controlează echipamentul este de-a dreptul îngrozitor şi utilizează doar 10% din performanţele plăcii.

Atenţie la extrapolarea rezultatelor

Să presupunem că s-au făcut măsurători ale unei anumite mărimi prin simularea încărcării reţelei între 0 (sistem complet descărcat) şi 0.4 (sistem încărcat în proporţie de 40%), conform punctelor de date şi liniilor continue dintre ele din fig. 6-42. Ar putea fi tentant să se extrapoleze liniar, aşa cum o sugerează linia punctată. Cu toate acestea, multe din rezultatele anterioare indică un factor de 1/(1 – p), unde p este încărcarea, astfel încât valorile adevărate pot arăta mai degrabă ca linia întreruptă, care creşte mult mai repede decât liniar.

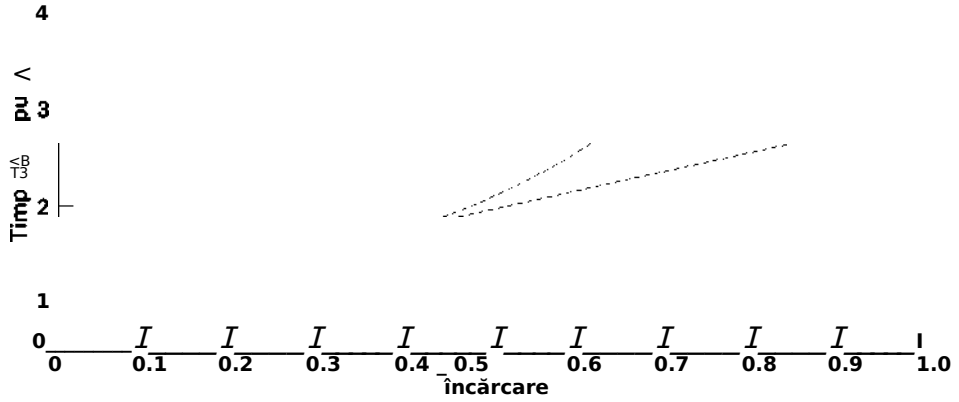


Fig. 6-42. Răspunsul, ca o funcţie de încărcare.

6.6.3 Proiectarea de sistem pentru performanţe superioare

Măsurarea şi ajustarea pot îmbunătăţi considerabil performanţele, dar ele nu pot

substitui o proiectare făcută bine de la început. O reţea proiectată superficial poate fi îmbunătăţită tot în aceeaşi măsură. O soluţie mai bună este să se refacă totul de la temelie.

În această secţiune, vom prezenta câteva reguli rezultate dintr-o experienţă acumulată pe un număr mare de reţele. Aceste reguli se referă nu doar la proiectarea de reţea, ci şi la proiectarea de sistem, şi aceasta pentru că programul şi sistemul de operare sunt deseori mai importante decât ruterul şi echipamentul de interfaţă. Cele mai multe din aceste idei constituiau cunoştinţe de bază ale proiectanţilor de reţele, care erau propagate din generaţie în generaţie pe cale orală. Ele au fost prima oară enunţate explicit de Mogul (1993); expunerea noastră merge în mare parte în paralel cu a sa. O altă sursă relevantă este (Metcalfe, 1993).

Regula #1: Viteza procesorului este mai importantă decât viteza reţelei.

O experienţă îndelungată a arătat că în aproape toate reţelele, supraîncărcarea indusă de sistemul de operare şi de protocol domină de fapt timpul de transmisie pe cablu. De exemplu, în teorie, timpul minim pentru un RPC pe o reţea Ethernet este de 102 ps, corespunzând unei cereri minime (64 de octeţi) urmate de un răspuns minim (64 de octeţi). În realitate, evitarea întârzierii suplimentare introduse de software şi obţinerea timpului RPC cât de cât apropiat de cel teoretic este o realizare considerabilă.

Similar, cea mai mare problemă în funcţionarea la 1 Gbps constă în plasarea biţilor din tamponul utilizatorului pe fibră suficient de repede şi în recepţionarea acestor biţi de către procesorul receptor la fel de repede cum sosesc. Pe scurt, dacă se dublează viteza procesorului, de regulă se poate ajunge până aproape de dublarea productivităţii. Dublarea capacităţii reţelei nu are de regulă nici un efect, deoarece gâtuirea se produce în general la calculatoarele gazdă.

Regula #2: Reducerea numărului de pachete pentru a reduce supraîncărcarea datorată programelor.

Prelucrarea unui TPDU adaugă o anumită supraîncărcare per TPDU (de exemplu prelucrarea antetului) şi o anumită cantitate de prelucrare suplimentară per octet (de exemplu calculul sumei de control). Atunci când se trimit un milion de octeţi, supraîncărcarea per octet este aceeaşi, indiferent dacă dimensiunea TPDU variază. Cu toate acestea, utilizarea de TPDU-uri de 128 octeţi presupune de 32 de ori mai multă supraîncărcare per TPDU faţă de cazul a 4K octeţi per TPDU. Această supraîncărcare creşte cu rapiditate.

În plus faţă de supraîncărcarea TPDU-ului, trebuie considerată şi supraîncărcarea datorată nivelurilor inferioare. Fiecare sosire a unui pachet generează o întrerupere. Pe un procesor cu bandă de asamblare modern, fiecare întrerupere fragmentează banda de asamblare a procesorului, interferă cu memoria tampon, presupune o schimbare de context în controlul memoriei şi impune salvarea unui număr important din registrele procesorului. O divizare prin n a numărului de TPDU-uri trimise reduce în consecinţă numărul de întreruperi şi supraîncărcarea pachetelor cu un factor de n.

Această observaţie justifică necesitatea colectării unei cantităţi importante de date înaintea transmisiei, în scopul reducerii numărului de întreruperi la celălalt capăt. Algoritmul Nagle şi soluţia Clark pentru sindromul ferestrei stupide reprezintă încercări de a face exact acest lucru.

Regula #3: Minimizarea numărului de comutări de context.

Comutările de context (de exemplu, din mod nucleu în mod utilizator) sunt catastrofale. Ele au aceleaşi proprietăţi incomode ca şi întreruperile, cea mai rea fiind o lungă serie de eşecuri la accesul iniţial la memoria ascunsă. Comutările de context pot fi reduse dacă funcţia de bibliotecă ce trimite date le stochează intern până la acumularea unei cantităţi semnificative. Similar, de partea receptorului, TPDU-urile de dimensiune mică recepţionate ar trebui colectate şi trimise utilizatorului dintrun singur foc şi nu individual, în scopul minimizării comutărilor de context.

În cel mai bun caz, sosirea unui pachet produce o comutare de context din modul utilizator curent în modul nucleu şi apoi o comutare către procesul receptor, astfel încât acesta din urmă să preia informaţia nou sosită. Din nefericire, în multe sisteme de operare se petrec comutări de contexte suplimentare. De exemplu, dacă procesul controlor de reţea rulează ca un proces special în spaţiul utilizator, sosirea unui pachet provoacă, probabil, o comutare de context de la utilizatorul curent către nucleu, apoi încă o comutare de la nucleu către controlorul de reţea, urmată de încă una înapoi către nucleu şi în final din nucleu către procesul receptor. Această secvenţă este prezentată de fig. 6-43. Toate aceste comutări de contexte pentru fiecare pachet consumă timp din timpul procesorului şi au un efect distrugător asupra performanţelor reţelei.

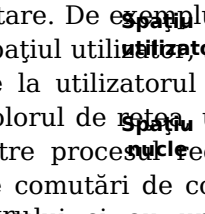


Fig. 6-43. Patru comutări de context pentru fiecare pachet, cu un controlor de reţea în

Regula #4: Minimizarea numărului de copieri.

Efectuarea unor copii multiple este şi mai dăunătoare decât comutările multiple de contexte. Nu este nimic deosebit în faptul că un pachet proaspăt recepţionat este copiat de trei sau patru ori înainte ca TPDU-ul conţinut în el să fie livrat. După ce un pachet este recepţionat de către echipamentul de reţea într-un tampon special cablat pe placă, el este copiat de obicei într-un tampon al nucleului. De aici el este copiat într-un tampon al nivelului reţea, apoi într-unul al nivelului transport şi în

final de către procesul aplicaţiei receptoare.

Un sistem de operare inteligent va copia câte un cuvânt odată, dar nu este deloc neobişnuit să fie necesare cinci instrucţiuni per cuvânt (încărcare, memorare, incrementarea unui registru index, un test pentru marcajul de sfârşit al datelor şi un salt condiţionat). A face trei copii ale fiecărui pachet la cinci instrucţiuni copiate pe fiecare cuvânt de 32 de biţi, necesită 15/4 sau aproximativ patru instrucţiuni per octet copiat. Pe un procesor de 500 MIPS, o instrucţiune durează 2 ns deci fiecare octet necesită 8 ns sau aproximativ 1 ns per bit, dând o rată de transfer de aproximativ 1 Gbps. Ţinând cont şi de prelucrarea antetului, tratarea întreruperilor şi a comutărilor de contexte, se pot atinge 500 Mbps, aceasta fără a pune la socoteală prelucrarea efectivă a datelor. Evident, controlul unei linii Ethernet de 10 Gbps care funcţionează la viteză maximă, nici nu intră în discuţie.

De fapt, probabil că nici o linie de 500 Mbps nu poate fi manipulată la viteză maximă. În calculul anterior am presupus că o maşină cu 500 MIPS poate executa oricare 500 milioane de instrucţiuni pe secundă. De fapt, maşinile pot opera la această viteză doar dacă nu execută referiri la memorie. Operaţiile cu memoria sunt, de cele mai multe ori, de zece ori mai lente decât instrucţiunile registru-registru (adică 20 ns / instrucţiune). Dacă 20 de procente din instrucţiuni chiar referă memoria (adică datele necesare nu se află în memoria tampon), ceea ce este de aşteptat când este vorba despre pachetele care vin, timpul mediu de execuţie este 5.6 ns (0.8x2+0.2x20). Cu patru instrucţiuni/octet, avem nevoie de 22.4 ns / octet, sau 2.8 ns / bit, care înseamnă aproximativ 357 Mbps. Considerând un procent de 50 supraîncărcare obţinem 178 Mbps. De notat că echipamentul suport nu ajută în acest caz. Problema este că sistemul de operare execută prea multe operaţii de copiere.

Regula #5: Oricând se poate cumpăra mai multă lărgime de bandă, dar niciodată o întârziere mai mică.

Următoarele trei reguli se ocupă de comunicaţie mai mult decât de prelucrarea protocolului. Prima regulă stabileşte că, dacă se doreşte o lăţime de bandă mai mare, este suficient să o cumperi. Dacă se pune o a doua fibră alături de prima, se dublează lărgimea de bandă, dar nu se micşorează deloc întârzierile. Micşorarea întârzierilor presupune îmbunătăţirea programului de protocol, a sistemului de operare sau a interfeţei cu reţeaua. Chiar dacă toate acestea sunt îndeplinite, întârzierea nu se va reduce dacă gâtuirea constă în timpul de transmisie.

Regula #6: Evitarea congestiei este preferabilă eliminării congestiei.

Vechea maximă conform căreia o uncie de prevenire este mai bună decât o livră de însănătoşire este cu certitudine valabilă şi în cazul congestiei reţelei. Atunci când o reţea este congestionată, se pierd pachete, lărgimea de bandă este irosită, apar întârzieri inutile şi multe altele. Recuperarea din congestie ia timp şi răbdare. Este de preferat să se procedeze de aşa natură, încât congestia să nu apară. Evitarea congestiei este ca şi vaccinarea împotriva tetanosului: doare puţin în momentul în care se face vaccinul, dar aceasta împiedică o durere mult mai mare mai târziu.

Regula #7: Evitarea întârzierilor.

În reţele sunt necesare contoare, dar ele ar trebui utilizate cu măsură şi întârzierile ar trebui minimizate. Atunci când expiră un contor, se repetă de regulă o acţiune. Dacă este într-adevăr necesară repetarea acţiunii respective, atunci asta este, dar repetarea ei fără rost reprezintă o risipă.

Modalitatea de a evita un efort suplimentar constă în înţelegerea faptului că aceste contoare se cam situează de partea conservatoare a lucrurilor. Un contor căruia îi ia mult timp ca să expire adaugă o mică întârziere suplimentară în cazul (puţin probabil) în care un TPDU se pierde. Un contor care expiră atunci când nu ar trebui, consumă în mod nepermis din timpul procesor, iroseşte din lărgimea de bandă şi adaugă o încărcare suplimentară în zeci de rutere, probabil fără nici un motiv serios.

6.6.4 Prelucrarea rapidă a TPDU-urilor

Morala poveştii anterioare este aceea că principalul obstacol către reţelele rapide îl constituie programul de protocol. În această secţiune, vom studia câteva modalităţi pentru a creşte viteza acestui program. Pentru mai multe informaţii, pot fi citite (Clark şi alţii 1989; şi Chase şi alţii, 2001).

Supraîncărcarea indusă de prelucrarea TPDU-urilor are două componente: supraîncărcare per TPDU şi supraîncărcare per octet. Ambele trebuie să fie abordate. Cheia accelerării prelucrării TPDU-urilor constă în separarea cazului normal (transferul datelor într-un singur sens) şi tratarea sa specială. Cu toate că este necesară o secvenţă de TPDU-uri speciale pentru a se intra într-o stare *STABILIT,* odată intrat în starea respectivă, prelucrarea TPDU-urilor decurge lin, până în clipa în care una din părţi iniţiază închiderea conexiunii.

Să începem examinarea părţii emiţătoare din starea *STABILIT*, atunci când există date de transmis. Pentru claritate, presupunem că entitatea transport este în nucleu, aceleaşi idei aplicându- se şi în cazul în care este vorba de un proces în spaţiul utilizator sau o bibliotecă în interiorul procesului emiţător. În fig. 6-44, pentru a executa SEND-ul, procesul emiţător intră în mod nucleu prin acţionarea unei capcane. Primul lucru pe care îl face entitatea transport este de a testa dacă nu cumva este vorba de cazul normal: starea este *STABILIT,* nici o parte nu încearcă să închidă conexiunea, un TPDU normal (adică unul care nu e în afara limitelor) este în curs de a fi transmis şi la receptor este disponibil un spaţiu fereastră suficient. Dacă toate condiţiile sunt îndeplinite, nici un test suplimentar nu mai este necesar şi poate fi acaparată calea rapidă prin entitatea de transport emiţătoare. De obicei, această cale este acaparată în majoritatea timpului.

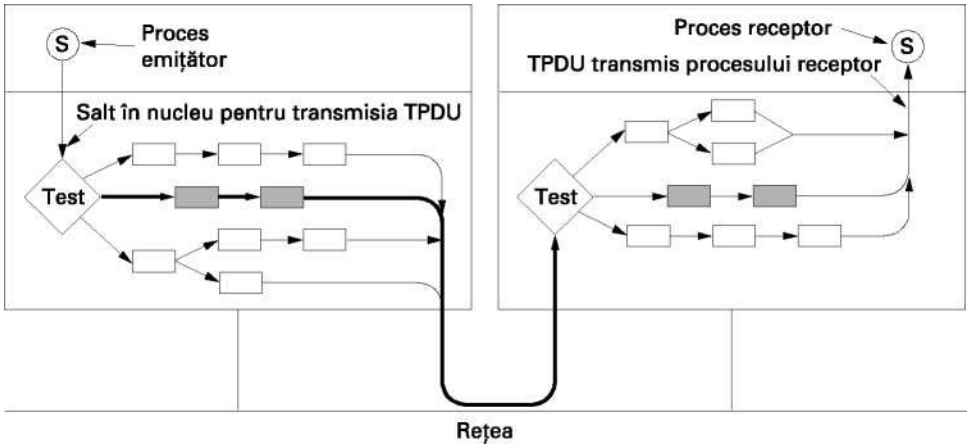


Fig. 6-44. Calea rapidă de la emiţător la receptor este indicată printr-o linie groasă. Paşii de prelucrare ai acestei căi sunt reprezentaţi prin dreptunghiuri umbrite.

În cazul obişnuit, antetele mai multor date TPDU consecutive sunt în mare parte identice. Pentru a profita de acest lucru, în interiorul entităţii transport se memorează un antet prototip. La începutul căii rapide, el este copiat cât de repede posibil într-un tampon special, cuvânt cu cuvânt. Acele câmpuri care ulterior se modifică de la un TPDU la altul sunt suprascrise în tampon. În mod frecvent, aceste câmpuri sunt uşor de derivat din variabilele de stare, de exemplu următorul număr de secvenţă. Apoi este pasat nivelului reţea un indicator spre întregul antet TPDU, împreună cu un indicator spre informaţia utilizator. Şi aici se poate urma aceeaşi strategie (caz neacoperit de fig. 6-44). În final, nivelul reţea furnizează pachetul rezultat nivelului legătură de date, în vederea transmisiei.

Pentru a exemplifica modul în care operează acest principiu în practică, să considerăm cazul TCP/IP-ului. Fig. 6-45(a) prezintă antetul TCP. Câmpurile haşurate sunt identice între două TPDU- uri consecutive, pe un flux într-un singur sens. Tot ce are de făcut entitatea transport este să copieze cele cinci cuvinte dintr-un antet prototip în tamponul care urmează să fie transmis, să completeze următorul număr de secvenţă (prin copierea lui dintr-un cuvânt din memorie), să calculeze suma de control şi să incrementeze numărul de secvenţă din memorie. Entitatea poate înmâna apoi antetul, împreună cu datele aferente, unei proceduri IP speciale, în vederea transmisiei obişnuite a unui TPDU de dimensiune maximă. În continuare IP copiază cele cinci cuvinte ale antetului său prototip [vezi fig. 6-45(b)] în tampon, completează câmpul *Identificare* şi calculează suma sa de control. Pachetul este acum gata pentru transmisie.

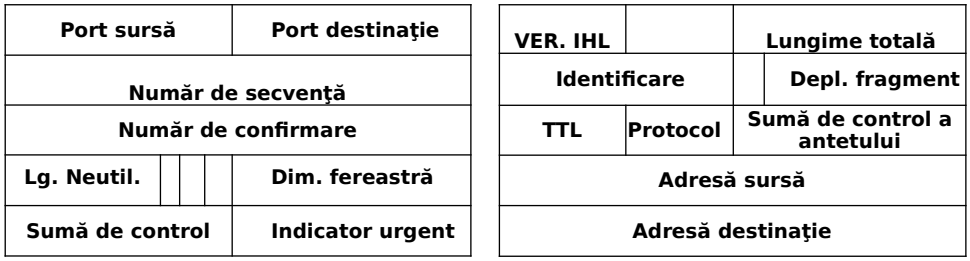


Fig. 6-45. (a) Antetul TCP. (b) Antetul IP. În ambele cazuri, câmpurile haşurate se obţin din prototip, fără nici o modificare.

Să aruncăm o privire asupra prelucrării pe calea rapidă în cazul receptorului din fig. 6-44. Pasul 1 constă în localizarea înregistrării de conexiune din TPDU-ul recepţionat. În cazul TCP, înregistrarea de conexiune poate fi memorată într-o tabelă de dispersie pentru care cheia poate fi o funcţie simplă aplicată celor două adrese IP şi celor două porturi. Odată ce înregistrarea de conexiune a fost localizată, corectitudinea sa trebuie verificată prin compararea ambelor adrese şi ambelor porturi.

O optimizare care accelerează şi mai mult determinarea înregistrării de conexiune constă în menţinerea unui indicator către ultima înregistrare utilizată, urmând ca aceasta să fie prima înregistrare testată. Clark ş.a. (1989) au aplicat această idee şi au observat o rată de succes care depăşeşte 90%. Alte euristici de căutare sunt descrise în (McKenney şi Dove, 1992).

În continuare, TPDU-ul este verificat pentru a determina dacă este vorba de cazul normal: starea este *STABILIT,* niciuna din părţi nu încearcă închiderea conexiunii, este un TPDU complet, nici un indicator special nu este poziţionat şi numărul de secvenţă este cel aşteptat. Aceste teste înseamnă doar câteva instrucţiuni. Dacă toate condiţiile sunt îndeplinite, este invocată o procedură TCP pentru cale rapidă.

Calea rapidă actualizează înregistrarea de conexiune şi copiază informaţia către utilizator. Totodată, suma de control este calculată chiar pe parcursul copierii, eliminând astfel trecerile suplimentare pe secvenţa de date. Dacă suma de control este corectă, înregistrarea de conexiune este actualizată şi se trimite o confirmare. Schema generală care constă într-un control rapid la început, pentru a vedea dacă antetul este cel aşteptat, precum şi în existenţa unei proceduri speciale care tratează cazul respectiv, se numeşte **predicţia antetului**. Schema este utilizată în multe din implementările TCP. Atunci când această optimizare este utilizată împreună cu celelalte optimizări discutate în acest capitol, este posibil ca TCP-ul să atingă la execuţie 90% din viteza de copiere locală memorie – memorie, presupunând că mediul de comunicaţie este suficient de rapid.

Alte două domenii unde se pot obţine câştiguri importante în performanţe sunt controlul tampoanelor şi al contoarelor de timp. În controlul tampoanelor, ideea constă în evitarea copierilor inutile, aşa cum s-a menţionat anterior. Controlul contoarelor de timp este important, deoarece aproape nici un contor nu expiră de fapt. Acestea sunt poziţionate astfel, încât să ne păzească împotriva pierderii de

TPDU-uri, dar majoritatea TPDU-urilor, ca şi confirmările lor, de altfel, ajung corect la destinaţie. Este deci important să se optimizeze controlul contoarelor de timp pentru cazul în care acestea expiră rar.

O schemă uzuală constă în utilizarea unei liste îmbunătăţite a evenimentelor generate de contoare, sortate în funcţie de momentul expirării acestora. Intrarea din capătul acestei liste conţine un contor care indică depărtarea în impulsuri de ceas faţă de momentul expirării. Fiecare din intrările care urmează indică printr-un contor depărtarea în impulsuri de ceas faţă de intrarea precedentă. Astfel, pentru evenimente care expiră în 3, 10 şi respectiv 12 impulsuri, cele trei contoare sunt 3, 7 şi, respectiv, 2.

La fiecare impuls de ceas, contorul din capătul listei este decrementat. Atunci când contorul atinge valoarea 0, se prelucrează evenimentul asociat lui şi următorul element din listă devine capătul listei. Contorul acestuia nu trebuie să fie modificat. Prin această schemă, inserarea şi ştergerea evenimentelor sunt operaţii costisitoare, cu un timp de execuţie proporţional cu lungimea listei.

Dacă intervalul maxim de expirare al contorului este limitat şi cunoscut în avans, poate fi utilizată o abordare mai eficientă. În acest caz, poate fi utilizat un vector numit **roata timpului**, ca în fig. 6-46. Fiecare poziţie corespunde unui impuls de ceas. Ceasul curent reprezentat este $T = 4$. Contoarele sunt planificate să expire la 3, 10 şi 12 impulsuri faţă de acest moment. Dacă un contor nou este brusc poziţionat să expire peste 7 impulsuri, se creează pur şi simplu o intrare în poziţia 11. Similar, în cazul în care contorul planificat să expire la $T + 10$ trebuie să fie anulat, trebuie parcursă lista care începe pe poziţia 14 şi intrarea cerută trebuie ştearsă. Să observăm că vectorul din fig. 6-46 nu poate suporta contoare care expiră după $T + 15$.

Indicator spre lista contoarelor pentru T +

Cu fiecare impuls de ceas indicatorul de timp curent este avansat cu o poziţie (circular). Dacă intrarea indicată este nenulă, sunt prelucrate toate contoarele asociate ei. Mai multe variaţiuni pe această temă sunt discutate în (Varghese şi Lauck, 1987).

6.6.5 Protocoale pentru reţele gigabit

La începutul anilor ‘90 au început să apară reţele gigabit. Prima reacţia a oamenilor a fost să utilizeze pentru ele vechile protocoale, lucru care a pus în scurt timp diferite probleme. Vom discuta în această secţiune câteva din aceste probleme, precum şi direcţiile urmate de noile protocoale pentru a le soluţiona, pe măsură ce evoluăm către reţele tot mai rapide.

Prima problemă este că multe protocoale utilizează secvenţe de numere de 32 de biţi. Când a apărut Internetul, liniile între rutere erau mai ales linii închiriate de 56 Kbps, astfel că unui calculator gazdă care mergea la viteză maximă îi trebuia mai mult de o săptămână să treacă prin toate numerele de secvenţă. Pentru proiectanţii de TCP, 2^{32} a fost o aproximaţie destul de decentă a infinitului, pentru că era mic pericolul ca pachete vechi să mai fie prin preajmă la o săptămână după ce au fost transmise. Cu o linie Ethernet de 10 Mbps, timpul de parcurgere a numerelor de secvenţă a devenit de 57 de minute, mult mai mic, dar încă administrabil. Cu o linie Ethernet de 1 Gbps transmiţând date prin Internet, timpul de parcurgere este de aproximativ 34 secunde, mult sub timpul maxim de viaţă al unui pachet în Internet, de 120 de secunde. Dintr-o dată, 2^{32} nu mai este o aproximare atât de bună a infinitului, din moment ce un transmiţător poate cicla prin spaţiul de secvenţă, în timp ce pachetele vechi există încă. RFC 1323 oferă totuşi o fereastră de ieşire.

Problema este că mulţi proiectanţi de protocoale au plecat de la presupunerea că timpul necesar consumării spaţiului numerelor de secvenţă depăşeşte cu mult timpul maxim de viaţă al unui pachet. În consecinţă, nu era nici măcar nevoie să îşi facă griji că duplicate vechi pot să existe încă undeva atunci când numerele de secvenţă au revenit la vechile valori. La viteze gigabit această presupunere cade.

O a doua problemă este aceea că viteza de comunicaţie a crescut mult mai repede decât viteza de prelucrare. (Notă pentru inginerii de calculatoare: Ieşiţi în stradă şi bateţi-i pe inginerii de comunicaţii! Ne bazăm pe voi.) În anii '70, ARPANET-ul opera la 56 Kbps şi avea calculatoare care funcţionau la aproape 1 MIPS. Pachetele erau de 1008 biţi şi astfel ARPANET-ul putea livra aproximativ 56 pachete/sec. Având disponibile 18 ms pentru fiecare pachet, o maşină gazdă putea să-şi permită să irosească 18000 instrucţiuni pentru prelucrarea unui pachet. Desigur, dacă ar fi făcut astfel, ar fi asfixiat complet procesorul, dar putea renunţa la doar 9000 instrucţiuni per pachet şi tot i-ar mai fi rămas jumătate din puterea procesorului pentru celelalte prelucrări.

Să comparăm aceste numere cu calculatoarele moderne de 1000 MIPS care interschimbă pachete de 1500 de octeţi pe o linie gigabit. Pachetele pot curge cu o viteză de peste 80000 pe secundă, astfel încât, dacă vrem să rezervăm jumătate din puterea procesorului pentru aplicaţii, prelucrarea unui pachet trebuie să se încheie în 6,25 psec. În 6,25 psec, un calculator de 1000 MIPS poate executa doar 6250

instrucţiuni, doar 1/3 din ceea ce îşi putea permite un calculator gazdă ARPANET. Mai mult decât atât, instrucţiunile RISC moderne fac mai puţine lucruri per instrucţiune decât o făceau vechile instrucţiuni CISC, deci problema este chiar mai gravă decât pare. În concluzie: există mult mai puţin timp pentru prelucrarea efectuată de protocol decât exista altădată, deci protocoalele trebuie să devină mai simple.

O a treia problemă este aceea că protocolul cu întoarcere în *n* paşi are performanţe slabe pe linii cu un produs lărgime de bandă-întârziere de valoare mare. Să considerăm de exemplu o linie de 4000 km operând la 1 Gbps. Timpul de transmisie dus-întors este de 40 ms, timp în care un emiţător poate transmite 5 megaocteţi. Dacă se detectează o eroare, atunci vor fi necesare 40 ms înainte ca emiţătorul să fie avertizat de acest lucru. Dacă este utilizat algoritmul cu întoarcere în *n* paşi, emiţătorul nu va avea de retransmis doar pachetul eronat, ci şi toţi cei 5 megaocteţi care au urmat după el. În mod clar, are loc o risipă mare de resurse.

O a patra problemă este aceea că liniile gigabit sunt fundamental diferite de liniile megabit, liniile gigabit de lungime mare fiind limitate de întârziere mai degrabă decât de lărgimea de bandă. În fig. 6-47 arătăm timpul necesar transferului unui fişier de un megabit la 4000 km distanţă pentru diferite viteze de transfer. La o viteză de până la 1 Mbps, timpul de transmisie este dominat de viteza la care pot fi transferaţi biţii. La 1 Gbps, întârzierea de 40 ms a circuitului dus-întors domină acea

milisecundă necesară pentru a pune bitul pe fir. Creşteri suplimentare ale lărgimii de bandă aproape că rămân fără efect.

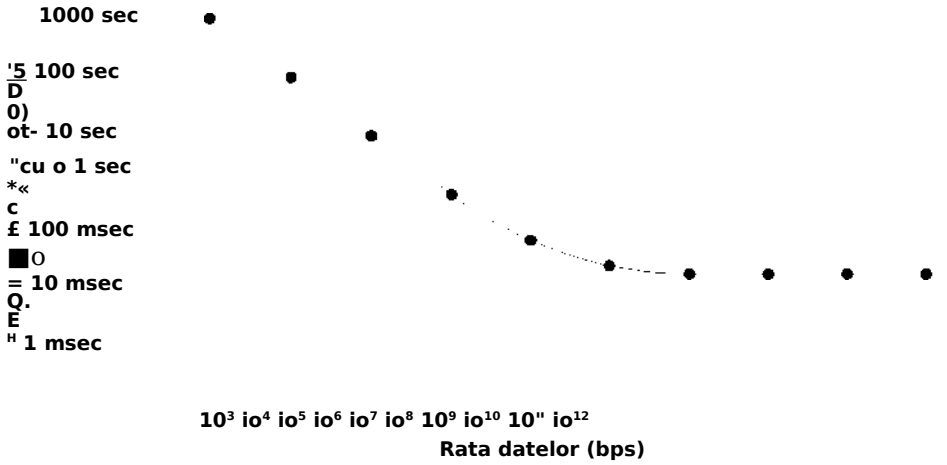


Fig. -47. Timpul necesar transferului şi confirmării unui fişier de un megabit

6

pe o linie de 4000 km.

Fig. 6-47 are implicaţii nedorite pentru protocoalele de reţea. Ea arată că protocoalele pas-cu- pas (stop-and-wait), precum RPC, au o limită superioară inerentă de performanţă. Această limită este dictată de viteza luminii. Oricât de mare ar fi progresul tehnologic în optică, nu se poate obţine nici o îmbunătăţire (deşi noi legi ale fizicii ar fi folositoare).

O a cincea problemă care merită menţionată nu este o problemă de protocol sau o problemă tehnologică, ci un rezultat al noilor aplicaţii. Enunţată doar, ea spune că pentru multe aplicaţii gigabit, precum multimedia, varianţa sosirii pachetelor în timp este la fel de importantă ca însăşi media întârzierilor. De cele mai multe ori este de preferat o viteză de livrare redusă, dar uniformă uneia rapide, dar fluctuante.

Să revenim acum de la identificarea problemelor la modalităţile lor de rezolvare. Vom face, pentru început, câteva remarci generale, apoi vom analiza mecanismele de protocol, aspectul pachetelor şi programele de protocol.

Principiul de bază pe care toţi proiectanţii de reţele gigabit ar trebui să-l înveţe pe de rost este: *Proiectaţi astfel încât să optimizaţi viteza, nu lăţimea de bandă.*

Protocoalele vechi au fost de regulă proiectate pentru a minimiza numărul de biţi de pe fir, prin utilizarea frecventă a câmpurilor de dimensiuni mici şi prin împachetarea lor în octeţi şi cuvinte. În zilele noastre, există suficient de multă lăţime de bandă. Problema o reprezintă prelucrările efectuate de protocoale, deci acestea trebuie reduse la minim. Clar că proiectanţii procesorului IPv6 au înţeles

acest principiu.

O modalitate tentantă pentru accelerarea lucrurilor este de a construi interfeţe rapide cu reţeaua sub formă de echipamente fizice. Dificultatea acestei strategii constă în aceea că, în lipsa unui protocol extrem de simplu, un nou echipament înseamnă conectarea unei noi plăci, cu un al doilea procesor şi cu propriul program. Pentru a asigura faptul că un co-procesor de reţea este mai ieftin decât procesorul principal, el este de regulă o componentă mai lentă. Consecinţa acestei soluţii este că mult timp, procesorul principal (rapid) este inactiv, aşteptând ca al doilea procesor (cel lent) să facă toată munca critică. Este un mit acela că procesorul principal ar avea altceva de făcut în acest timp. Mai mult decât atât, atunci când comunică două procesoare independente, pot apărea condiţii de cursă, fiind deci necesare protocoale complexe, pentru sincronizarea lor corectă. În general, cea mai bună abordare este de a concepe protocoalele cât mai simplu şi de a lăsa procesorul principal să facă toată treaba.

Să studiem acum noţiunea de reacţie în protocoalele de mare viteză. Datorită buclei de întârziere (relativ) lungă, reacţia ar trebui evitată: receptorului îi ia prea mult timp pentru a anunţa emiţătorul. Un exemplu de reacţie este controlul vitezei de transfer prin utilizarea unui protocol cu fereastră glisantă. Pentru a evita întârzierile (îndelungate) inerente atunci când receptorul trimite emiţătorului actualizările de fereastră, cel mai bine este să se utilizeze o rată de transfer dictată de protocol. Într-un astfel de protocol, emiţătorul poate trimite tot ceea ce doreşte el să trimită, cu condiţia să nu o facă mai repede decât s-a convenit iniţial cu receptorul.

Un al doilea exemplu de reacţie este algoritmul startului lent al lui Jacobson. Acest algoritm face interogări multiple pentru a determina cât de mult poate suporta reţeaua. În reţelele de mare viteză utilizarea a jumătate de duzină de interogări scurte pentru testarea răspunsul reţelei conduce la irosirea unei părţi imense din lăţimea de bandă. O schemă mai eficientă este ca emiţătorul, receptorul şi reţeaua să rezerve resursele necesare în momentul stabilirii conexiunii. Rezervarea în avans a resurselor are de asemenea avantajul diminuării fluctuaţiilor. Pe scurt, migrarea spre viteze mari împinge inexorabil proiectarea spre operaţii orientate pe conexiuni sau spre ceva foarte apropiat de acestea. Desigur, dacă lărgimea de bandă nu va mai fi o problemă în viitor încât nimănui să nu-i mai pese de pierderi, regulile de proiectare vor deveni foarte diferite.

În reţelele gigabit, o importanţă deosebită trebuie acordată organizării pachetelor. Antetul ar trebui să conţină cât mai puţine câmpuri cu putinţă pentru a reduce timpul de prelucrare, iar aceste câmpuri ar trebui să fie suficient de mari pentru a-şi îndeplini scopul şi pentru a fi aliniate la cuvânt, fiind astfel mai uşor de prelucrat. În acest context, „suficient de mari” înseamnă eliminarea problemelor precum revenirea numerelor de secvenţă la valori vechi cât timp există încă pachete vechi, incapacitatea receptorilor de a oferi suficient spaţiu de fereastră datorită unei dimensiuni prea mici a câmpului fereastră ş.a.m.d.

Antetul şi informaţia ar trebui să fie prevăzute cu sume de control separate, din două motive. În primul rând, pentru a face posibilă calcularea sumei de control a antetului, dar nu şi a datelor. În al doilea rând, pentru a determina corectitudinea

antetului înaintea începerii copierii datelor în spaţiul utilizator. Este de dorit să se calculeze suma de control a datelor la momentul copierii lor în spaţiul utilizator, dar dacă antetul este incorect, copierea poate să se facă în spaţiul unui alt proces. Pentru a evita o copiere incorectă, dar pentru a permite şi calculul sumei de control a datelor în timpul copierii, este esenţial ca cele două sume de control să fie separate.

Dimensiunea maximă a datelor ar trebui să fie mare pentru a permite operaţii eficiente chiar şi în situaţia transferului la distanţe mari. De asemenea, cu cât este mai mare blocul de informaţie, cu atât este mai mică fracţiunea din totalitatea lăţimii de bandă alocată antetului. 1500 de octeţi este prea puţin.

O altă caracteristică importantă este posibilitatea de a trimite o cantitate rezonabilă de informaţie o dată cu cererea de conexiune. În acest fel se poate salva un timp de comunicaţie dus-întors.

În sfârşit, sunt utile câteva cuvinte despre programul de protocol. Cea mai mare atenţie trebuie acordată cazului de succes. Multe din protocoalele vechi aveau tendinţa de a evidenţia ce este de făcut atunci când ceva nu mergea cum trebuie (de exemplu pierderea unui pachet). Pentru a face protocoalele să meargă mai repede, proiectanţii ar trebui să aibă ca scop minimizarea timpului de prelucrare atunci când totul funcţionează corect. Minimizarea timpului de prelucrare în caz de eroare trebuie să treacă pe planul doi.

Un al doilea aspect legat de programe este minimizarea timpului de copiere. Aşa cum am văzut mai devreme, copierea datelor introduce în general un cost suplimentar. Ideal ar fi ca echipamentul fizic să depoziteze în memorie fiecare pachet recepţionat ca un bloc contiguu de date. Programul ar trebui apoi să copieze acest pachet în tamponul utilizator printr-o singură copiere de bloc. În funcţie de modul de lucru al memoriei tampon, ar fi de dorit chiar să se evite copierea în buclă. Cu alte cuvinte, cel mai rapid mod de a copia 1024 de cuvinte este de a avea 1024 de instrucţiuni MOVE una după cealaltă (sau 1024 de perechi încărcare-memorare). Rutina de copiere este critică într-o asemenea măsură, încât, dacă nu există altă modalitate de a păcăli compilatorul ca să producă cu precizie codul optimal, ea ar trebui scrisă de mână, cu multă atenţie, direct în cod de asamblare.

6.7 REZUMAT

Nivelul transport reprezintă cheia pentru înţelegerea protocoalelor organizate pe niveluri. El furnizează diferite servicii, cel mai important dintre acestea fiind fluxul de octeţi capăt-la-capăt de la emiţător la receptor, fiabil şi orientat pe conexiuni. El este accesat prin primitive de serviciu care permit stabilirea, utilizarea şi eliberarea conexiunilor. O interfaţă de nivel de transport obişnuită este cea oferită de soclurile Berkeley.

Protocoalele de transport trebuie să fie capabile să controleze conexiunea în reţele nefiabile. Stabilirea conexiunii este complicată de existenţa pachetelor duplicate întârziate, care pot apărea la momente inoportune. Pentru a le face faţă,

stabilirea conexiunii trebuie făcută prin intermediul protocoalelor cu înţelegere în trei paşi. Eliberarea unei conexiuni este mai simplă decât stabilirea sa, dar este încă departe de a fi banală datorită problemei celor două armate.

Chiar şi în cazul unui nivel reţea complet fiabil, nivelul transport are suficient de mult de lucru. El trebuie să controleze toate primitivele de serviciu, toate conexiunile şi contoarele de timp şi trebuie să aloce şi să utilizeze credite.

Internetul are două protocoale de transport principale: UDP şi TCP. UDP este un protocol neorientat pe conexiune care este în principal un ambalaj pentru pachetele IP, cu caracteristicile suplimentare de multiplexare şi demultiplexare a proceselor multiple folosind o singura adresa de IP. UDP poate fi folosit pentru interacţiunile client-server, de exemplu, utilizând RPC. De asemenea poate fi folosit pentru construirea protocoalelor în timp real cum ar fi RTP.

Principalul protocol de transport în Internet este TCP. El oferă un flux sigur, bidirecţional de octeţi. El utilizează un antet de 20 de octeţi pentru toate segmentele. Segmentele pot fi fragmentate de rutere în cadrul Internet-ului, deci calculatoarele gazdă trebuie să fie pregătite să le reasambleze. S-a depus un mare efort pentru optimizarea performanţelor TCP-ului, utilizând algoritmii propuşi de Nagle, Clark, Jacobson, Karn şi alţii. Legăturile fără fir adaugă o varietate de complicaţii TCP-ului.

TCP Tranzacţional este o extensie a TCP-ului care se ocupă de interacţiunile client-server cu un număr redus de pachete.

Performanţele reţelei sunt dominate în mod tipic de protocol şi de costul suplimentar datorat tratării TPDU-urilor, situaţie care se înrăutăţeşte la viteze mari. Protocoalele ar trebui proiectate astfel, încât să minimizeze numărul de TPDU-uri, copierile lor repetate şi comutările de context. Pentru reţelele gigabit sunt de dorit protocoale simple.

6.8 PROBLEME

1. În exemplele noastre de primitive de transport din fig. 6-2, LISTEN este un apel blocant. Este acest lucru strict necesar? Dacă nu, explicaţi cum ar putea fi utilizată o primitivă neblocantă. Ce avantaje ar avea aceasta pentru schema descrisă în text?

2. În modelul pe care se bazează fig. 6-4 se presupune că pachetele pot fi pierdute de către nivelul reţea şi trebuie deci să fie confirmate individual. Să presupunem că nivelul reţea este 100% fiabil şi nu pierde pachete niciodată. Ce modificări sunt necesare (dacă sunt necesare) în fig. 6-4?

3. În ambele părţi ale fig. 6-6 este un comentariu conform căruia valoarea SERVER_PORT trebuie să fie aceeaşi şi în client şi în server. De ce este acest lucru atât de important?

4. Să presupunem că pentru generarea numerelor de secvenţă iniţiale se utilizează o schemă dirijată de ceas cu un contor de timp de 15 biţi. Ceasul generează un impuls la fiecare 100 ms şi durata de viaţă maximă a unui pachet este de 60 sec. Cât de des este necesar să aibă loc o resincronizare
a) în cel mai rău caz?
b) atunci când se consumă 240 de numere de secvenţă pe secundă?

5. De ce este necesar ca timpul maxim de viaţă al unui pachet, *T,* să fie suficient de mare pentru a acoperi nu numai dispariţia pachetului, dar şi a confirmării?

6. Să ne imaginăm că pentru stabilirea unei conexiuni se utilizează un protocol cu înţelegere în doi paşi şi nu unul cu înţelegere în trei paşi. Cu alte cuvinte, al treilea mesaj nu mai este necesar. Sunt posibile interblocări în această situaţie? Daţi un exemplu sau arătaţi că nu există nici o interblocare.

7. Imaginaţi o problemă generalizată a celor *n* armate, în care acordul dintre oricare două armate este suficient pentru victorie. Există un protocol care îi permite albastrului să câştige?

8. Să considerăm problema recuperării după defectarea unei maşini gazdă (adică fig. 6-18). Dacă intervalul dintre scrierea şi trimiterea unei confirmări, sau vice-

versa, poate fi făcut relativ scurt, care sunt cele mai bune strategii emiţător-receptor pentru minimizarea şansei de defectare a protocolului?

9. Sunt posibile interblocările pentru entităţile transport descrise în text (fig. 6-20)?

10. Din pură curiozitate, programatorul entităţii transport din fig. 6-20 a decis să pună contoare în interiorul procedurii *sleep,* pentru a colecta astfel statistici despre vectorul *conn.* Între acestea se află şi numerele de conexiuni din fiecare dintre cele şapte stări posibile *Yn* (*i*=1,..7). După scrierea unui program FORTRAN serios pentru a analiza datele, programatorul nostru descoperă că relaţia $Ln_i = MAX\_CONN$ pare să fie totdeauna adevărată. Există şi alţi invarianţi care să implice doar aceste şapte variabile?

11. Ce se întâmplă dacă utilizatorul entităţii transport din fig. 6-20 trimite un mesaj de lungime 0? Discutaţi semnificaţia răspunsului.

12. Pentru fiecare eveniment care poate apărea în entitatea transport din fig. 6-20, spuneţi dacă este sau nu ca utilizatorul să doarmă (eng.: sleep) în starea *transmisie.*

13. Discutaţi avantajele şi dezavantajele creditelor faţă de protocoalele cu fereastră glisantă.

14. De ce există UDP? Nu ar fi fost suficient ca procesul utilizator să fie lăsat să trimită pachete bloc IP?

15. Se consideră un protocol simplu la nivelul aplicaţiei, construit pe UDP, care permite unui client să recupereze un fişier de la un server la distanţă aflat la o adresă bine cunoscută. Clientul trimite mai întâi o cerere cu numele fişierului, iar serverul răspunde cu o secvenţă de pachete de date conţinând diferite părţi din fişierul cerut. Pentru a asigura fiabilitate şi livrare secvenţială, clientul şi serverul folosesc un protocol pas-cu-pas. Ignorând problema evidentă a performanţei, vedeţi vreo problemă la acest protocol? Gândiţi-vă atent la posibilitatea terminării bruşte a proceselor.

16. Un client trimite cereri de 128 de octeţi către un server localizat la 100 km depărtare, printr-un cablu optic de 1 gigabit. Care este eficienţa liniei în timpul apelului de procedură la distanţă?

17. Să considerăm din nou situaţia din problema precedentă. Calculaţi timpul minim posibil de răspuns atât pentru linia de 1 Gbps anterioară cât şi pentru o linie de 1 Mbps. Ce concluzie puteţi trage?

18. Atât UDP, cât şi TCP, folosesc numere de port pentru a identifica entitatea destinaţie când livrează mesaje. Daţi două motive pentru care aceste protocoale au inventat un nou ID abstract (numere de port), în loc să folosească ID-uri de procese, care existau deja când aceste protocoale au fost proiectate.

19. Care este dimensiunea totală a MTU TCP, incluzând supraîncărcarea TCP-ului

şi IP-ului dar neincluzând supraîncărcarea nivelului legătură de date?

20. Fragmentarea şi reasamblarea datagramelor sunt controlate de IP şi sunt invizibile TCP-ului. Înseamnă acest lucru că TCP-ul nu trebuie să-şi facă griji pentru datele care sosesc în ordine eronată?

21. RTP este utilizat pentru a transmite date audio de calitatea CD-ului, ceea ce înseamnă o pereche de eşantioane pe 16 biţi de 44.100 de ori pe secundă, un eşantion pentru fiecare dintre canalele stereo. Câte pachete pe secundă trebuie să transmită RTP?

22. Ar fi posibil să fie plasat cod RTP în nucleul sistemului de operare, alături de cod UDP? Explicaţi răspunsul.

23. Un proces de pe maşina 1 a fost asociat portului *p* şi un proces de pe maşina 2 a fost asociat portului *q.* Este posibil ca între cele două porturi să fie deschise mai multe conexiuni TCP în acelaşi timp?

24. În fig. 6 -29 am văzut că alături de câmpul *Acknowledgement* de 32 de biţi, este un bit *ACK* în al patrulea cuvânt. Acesta adaugă cu adevărat ceva? De ce da, sau de ce nu?

25. Informaţia utilă maximă dintr-un segment TCP este de 65495 octeţi. De ce a fost ales un număr atât de straniu?

26. Descrieţi două moduri de a ajunge în starea *SYN RCVD* din fig. 6-28.

27. Prezentaţi un dezavantaj potenţial al algoritmului Nagle atunci când este utilizat într-o reţea puternic congestionată.

28. Să considerăm efectul utilizării startului lent pe o linie cu timpul circuitului dus-întors de 10 ms şi fără congestie. Fereastra receptorului este de 24 Kocteţi şi dimensiunea maximă a segmentului este de 2 Kocteţi. Cât timp trebuie să treacă înainte ca prima fereastră completă să poată fi trimisă?

29. Să presupunem că fereastra de congestie TCP este de 18 Kocteţi şi apare o depăşire de timp. Cât de mare va fi fereastra dacă următoarele patru rafale de transmisie reuşesc? Se presupune că dimensiunea maximă a segmentului este de 1 Koctet.

30. Dacă timpul circuitului TCP dus-întors, RTT, este la un moment dat 30 ms şi următoarele confirmări sosesc după 26, 32 şi, respectiv, 24 ms, care este noul RTT estimat folosind algoritmul Jacobson? Utilizaţi a=0.9.

31. O maşină TCP trimite cadre de 65535 octeţi pe un canal de 1 Gbps pentru care întârzierea pe un singur sens este de 10 ms. Care este productivitatea maximă care poate fi atinsă? Care este eficienţa liniei?

32. Care este cea mai mare viteză a liniei la care o maşină gazdă poate transmite pachete TCP de 1500 octeţi cu un timp de viaţă care poate fi de maxim 120 sec fără ca numărul de secvenţă să se repete? Luaţi în considerare supraîncărcarea

de la TCP, IP şi Ethernet. Presupuneţi că se pot transmite continuu cadrele Ethernet.

33. Într-o reţea care are dimensiunea maximă a TPDU-urilor de 128 octeţi, timpul maxim de viaţă al unui TPDU de 30 sec şi numărul de secvenţă de 8 biţi, care este rata maximă de date per conexiune?

34. Să presupunem că măsuraţi timpul necesar recepţionării unui TPDU. Atunci când apare o întrerupere, se citeşte timpul sistem în milisecunde. Când TPDU-ul este complet prelucrat, se citeşte din nou timpul sistem. S-au înregistrat 0 ms de 270000 de ori şi 1 de 730000 de ori. Care este timpul de recepţie al unui TPDU?

35. Un procesor execută instrucţiuni la o viteză de 1000 MIPS. Informaţia poate fi copiată câte 64 de biţi odată, fiecare copiere a unui cuvânt costând zece instrucţiuni. Dacă un pachet recepţionat trebuie să fie copiat de patru ori, poate sistemul să facă faţă unei linii de 1 Gbps? Pentru a simplifica, presupunem că toate instrucţiunile, inclusiv acelea de citire/scriere din memorie, rulează la viteza maximă de 1000 MIPS.

36. Pentru a evita problema revenirii numerelor de secvenţă la valori iniţiale în timp ce există încă pachete vechi, s-ar putea utiliza numere de secvenţă pe 64 de biţi. Cu toate acestea, teoretic, un cablu optic poate opera la 75 Tbps. Care este durata maximă de viaţă pe care trebuie să o aibă un pachet pentru ca viitoarele reţele de 75 Tbps să nu se lovească de aceeaşi problemă a revenirii numerelor de secvenţă chiar şi în cazul reprezentării lor pe 64 biţi? Presupuneţi, ca şi TCP- ul, că fiecare octet are propriul său număr de secvenţă.

37. Daţi un avantaj al RPC pe UDP faţă de TCP tranzacţional. Daţi un avantaj al T/TCP faţă de RPC.

38. În fig. 6-40(a), vedem că sunt necesare 9 pachete pentru a realiza RPC-ul. Există situaţii în care sunt necesare exact 10 pachete?

39. În secţiunea 6.6.5 am calculat că o linie gigabit livrează unei maşini gazdă 80000 de pachete pe secundă, permiţându-i doar 6250 de instrucţiuni pentru a prelucra un pachet şi lăsând doar jumătate din capacitatea procesorului pentru aplicaţii. Acest calcul presupune un pachet de 1500 octeţi. Refaceţi calculul pentru pachetele ARPANET de dimensiune de 128 octeţi. În ambele cazuri, presupuneţi că dimensiunile pachetelor date includ toate supraîncărcările.

40. Pentru o reţea care operează la 1 Gbps pe o distanţă de 4000 km, factorul limitator nu este dat de lărgimea de bandă, ci de întârziere. Considerăm un MAN cu sursa şi destinaţia situate în medie la 20 km una de cealaltă. La ce viteză de date întârzierea circuitului dus-întors datorată vitezei luminii egalează întârzierea de transmisie pentru un pachet de 1 Koctet?

41. Calculaţi produsul dintre întârziere şi lăţimea de bandă pentru următoarele reţele: (1) T1 (1,5 Mbps), (2) Ethernet (10 Mbps), (3) T3 (45 Mbps) şi (4) STS-3

(155 Mbps). Presupuneţi RTT de 100 ms. Amintiţi-vă ca antetul TCP-lui are 16 biţi rezervaţi pentru Dimensiunea Ferestrei. Care sunt implicaţiile din punctul de vedere al calculelor voastre?

42. Care este produsul dintre întârziere şi lăţimea de bandă pentru un canal de 50 Mbps pe un satelit geostaţionar? Dacă pachetele sunt toate de 1500 de octeţi (incluzând supraîncărcarea), cât de mare ar trebui să fie fereastra în pachete?

43. Serverul de fişiere din fig. 6-6 este departe de a fi perfect şi i-ar fi folositoare câteva îmbunătăţiri. Faceţi următoarele modificări:
a) Daţi clientului un al treilea argument care specifică un interval de octeţi.
b) Adăugaţi un flag -w de client care permite fişierului să fie scris pe server.

44. Modificaţi programul din fig. 6-20 pentru a asigura revenirea din erori. Adăugaţi un nou tip de pachet, *reset,* care poate ajunge doar după ce conexiunea a fost deschisă de ambele părţi şi n-a fost închisă de niciuna. Acest eveniment, care are loc simultan la ambele capete ale conexiunii, indică faptul că orice pachet care era în tranzit a fost sau distrus, sau livrat, în orice caz el nemaiaflându-se în subreţea.

45. Scrieţi un program care simulează controlul tampoanelor într-o entitate transport, utilizând un flux de control cu fereastră glisantă şi nu un control al fluxului cu credite, ca în fig. 6-20. Lăsaţi procesele de pe nivelul superior să deschidă conexiuni, să trimită date şi să închidă conexiuni în mod aleatoriu. Pentru a păstra programul cât mai simplu, faceţi ca toată informaţia să călătorească doar de la maşina A la maşina B şi deloc în sens invers. Experimentaţi cu diferite strategii de alocare a tampoanelor la nivelul maşinii B, ca, de exemplu, tampoane dedicate unei anume conexiuni faţă de tampoane preluate dintr-un depozit comun şi măsuraţi productivitatea totală atinsă în ambele cazuri.

46. Proiectaţi şi implementaţi un sistem de discuţii care permite mai multor grupuri de utilizatori să discute. Coordonatorul discuţiilor se află la o adresă de reţea bine cunoscută, foloseşte UDP pentru comunicarea cu clienţii de discuţii, setează serverele de discuţii pentru fiecare sesiune de discuţii, şi menţine un director de sesiuni de discuţii. Este un server de discuţii pentru fiecare sesiune de discuţii. Un server de discuţii foloseşte TCP pentru comunicaţie cu clienţi. Un client de discuţii permite utilizatorilor să pornească, să intre sau să iasă dintr-o sesiune de discuţii. Proiectaţi şi implementaţi codul pentru coordonator, server şi client.

NIVELUL APLICAŢIE

După ce am epuizat toate preliminariile putem aborda nivelul unde pot fi găsite toate aplicaţiile. Nivelurile de sub nivelul aplicaţie servesc la asigurarea unui transport sigur, dar nu îndeplinesc nici o funcţie concretă pentru utilizatori. În acest capitol vom studia câteva aplicaţii reale.

Totuşi, chiar şi la nivelul aplicaţie, apare necesitatea unor protocoale-suport care să permită funcţionarea aplicaţiilor reale. Înainte de a începe studiul aplicaţiilor, ne vom ocupa de unul dintre acestea. Subiectul în discuţie este DNS, care se ocupă de convenţiile de nume în Internet. După aceea vom examina trei aplicaţii reale: poşta electronică, World Wide Web (rom.: reţea de întindere planetară) şi, în final, multimedia.

7.1 DNS – SISTEMUL NUMELOR DE DOMENII

Cu toate că teoretic programele ar putea să se refere la sistemele gazdă, la cutiile poştale şi la alte resurse prin adresa lor de reţea (de exemplu prin adresa IP), aceste adrese sunt greu de memorat de către oameni. De asemenea, în trimiterea de poştă electronică la *tana@128.111.24.41* ar însemna că dacă furnizorul de servicii Internet sau organizaţia Tanei mută serverul de poştă pe o maşină diferită, cu o adresă IP diferită, adresa ei de e-mail se va schimba. De aceea au fost introduse nume ASCII pentru a separa numele maşinilor de adresele maşinilor. În acest fel,

adresa Tanei ar putea fi ceva de genul *tana@art.ucsb.edu*. Cu toate acestea, reţeaua înţelege numai adrese numerice, deci este necesar un mecanism care să convertească şirurile ASCII în adrese de reţea. În secţiunile următoare se va studia cum este realizată această conversie în Internet.

Încă de la ARPANET exista un fişier *host.txt* care cuprindea toate sistemele gazdă şi adresele lor IP. În fiecare noapte, toate gazdele îl preluau de la situl unde era păstrat. Pentru o reţea formată din câteva sute de maşini mari, cu divizarea timpului, această abordare era destul de rezonabilă.

Totuşi, atunci când la reţea au fost conectate mii de staţii de lucru, toţi şi-au dat seama că această abordare nu putea să funcţioneze la nesfârşit. În primul rând dimensiunea fişierului ar deveni prea mare. Cu toate acestea şi chiar mai important, conflictele de nume de sisteme gazdă ar apărea în permanenţă dacă nu ar fi administrate centralizat, ceva de negândit într-o reţea internaţională de dimensiuni uriaşe din cauza încărcării şi a latenţei. Pentru a rezolva aceste probleme, a fost inventat **DNS** (**Domain Name System –** Sistemul numelor de domenii).

Esenţa DNS-ului constă într-o schemă ierarhică de nume de domenii şi a unui sistem de baze de date distribuite pentru implementarea acestei scheme de nume. În principal este utilizat pentru a pune în corespondenţă numele sistemelor gazdă şi adresele destinaţiilor de e-mail cu adresele IP, dar poate fi utilizat şi pentru alte scopuri. DNS este definit în RFC-urile 1034 şi 1035.

Foarte pe scurt, DNS este utilizat după cum urmează. Pentru a stabili corespondenţa dintre un nume şi o adresă IP, programul de aplicaţie apelează o procedură de bibliotecă numită **resolver**, transferându-i numele ca parametru. Putem vedea un exemplu de resolver, *gethostbyname,* în fig. 66. Resolver-ul trimite un pachet UDP la serverul DNS local, care caută numele şi returnează adresa IP către resolver, care o returnează apelantului. Înarmat cu adresa IP, programul poate stabili o conexiune TCP cu destinaţia sau îi poate trimite pachete UDP.

7.1.1 Spaţiul de nume DNS

Administrarea unui volum mare de nume în permanentă schimbare nu este o problemă prea uşoară. În sistemul poştal, administrarea numelor este realizată impunând ca pe o scrisoare să se specifice (implicit sau explicit) ţara, statul sau provincia, oraşul, strada şi restul adresei destinatarului. Utilizând o astfel de adresare ierarhică, nu există nici o confuzie între Marvin Anderson de pe Main St. din White Plains, N.Y. şi Marvin Anderson de pe Main St. din Austin, Texas. DNS lucrează în acelaşi mod.

Conceptual, Internetul este divizat în peste 200 **domenii** de nivel superior, fiecare domeniu cuprinzând mai multe sisteme gazdă. Fiecare domeniu este partiţionat în subdomenii şi acestea sunt, la rândul lor, partiţionate ş.a.m.d. Toate aceste domenii pot fi reprezentate ca un arbore, aşa cum se arată în fig. 7-1. Frunzele arborelui reprezintă domenii care nu au subdomenii (dar, bineînţeles, conţin sisteme). Un domeniu frunză poate conţine un singur sistem gazd ă sau poate reprezenta o firmă, deci să conţină mii de sisteme gazdă.

Domeniile de pe primul nivel se împart în două categorii: generice şi de ţări.

Domeniile generice sunt *com* (comercial), *edu* (instituţii educaţionale), *gov* (guvernul federal al SUA), *int* (organizaţii internaţionale), *mil* (forţele armate ale SUA), *net* (furnizori Internet) şi *org* (organizaţii nonprofit). Domeniile de ţări includ o intrare pentru fiecare ţară, cum se defineşte în ISO 3166.

În noiembrie 2000, ICANN a aprobat patru domenii de nivel superior noi, de interes general, şi anume, *biz* (afaceri), *info* (informaţii), *name* (nume de persoane), şi *pro* (profesii, ca de exemplu doctori sau avocaţi). În plus, au fost introduse trei domenii de nivel superior cu caracter specializat, cerute de către anumite industrii. Acestea sunt *aero* (industria aerospaţială), *coop* (cooperative), şi *museum* (muzee). În viitor vor fi adăugate alte domenii superioare.

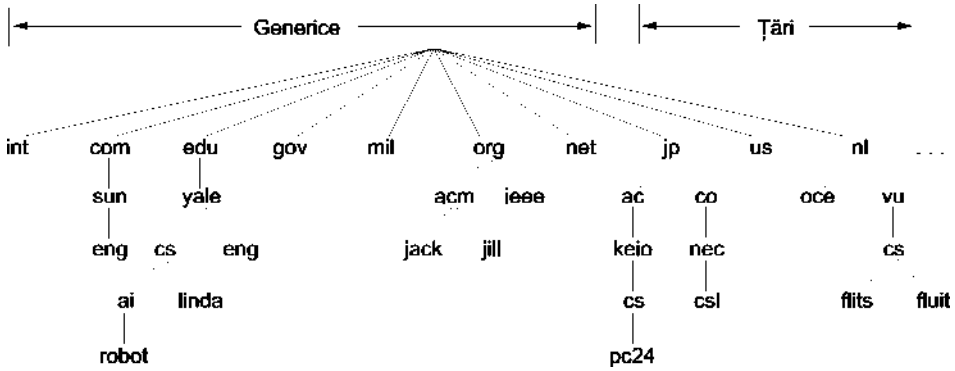


Fig. 7-1. O porţiune a spaţiului numelor de domenii din Internet.

Pe de altă parte, pe măsură ce Internetul devine mai comercial, el devine şi mai discutabil. Să luăm domeniul *pro,* de exemplu, care a fost proiectat pentru profesioniştii atestaţi. Dar cine este un profesionist? Şi de cine este atestat? Dar cum rămâne cu fotografii, profesorii de pian, magicienii, instalatorii, frizerii, exterminatorii, artiştii de tatuaje, mercenarii şi prostituatele? Sunt acestea meserii şi sunt acceptabile pentru domeniile *pro*? Şi daca da, cine atestă diverşi practicieni?

În general, obţinerea unui domeniu de nivel secundar, ca de exemplu *nume-al-companiei.com,* este uşoară. Pur şi simplu este necesară doar consultarea serviciului de înregistrare al nivelului superior corespunzător (*com* în acest caz) pentru a vedea dacă numele dorit este disponibil şi nu aparţine altcuiva. Dacă nu sunt probleme, solicitantul plăteşte o mică taxă anuală şi primeşte numele. Până acum, cam toate cuvintele comune (din engleză) au fost luate în domeniul *com*. Încercaţi nume de articole casnice, animale, plante, părţi ale corpului etc. Aproape toate sunt luate.

Fiecare domeniu este identificat prin calea în arbore de la el la domeniul (fără nume) rădăcină. Componentele sunt separate prin puncte (pronunţat „dot”). Astfel, departamentul tehnic de la Sun Microsystems ar putea fi *eng.sun.com,* în loc de numele în stil UNIX */com/sun/eng.* De notat că această numire ierarhică face ca *eng.sun.com* să nu intre în conflict cu posibila utilizare a lui *eng* din *eng.yale.edu,* care ar putea fi folosit pentru departamentul de limba engleză de la Yale.

Numele de domenii pot fi absolute sau relative. Un nume absolut de domeniu se termină cu un punct (de exemplu, *eng.sun.com.),* în timp ce unul relativ nu. Numele

relative trebuie interpretate în context pentru a le determina înţelesul adevărat. În ambele cazuri, un nume de domeniu se referă la un anumit nod din arbore şi la toate nodurile de sub el.

Numele de domenii nu fac distincţie între litere mici şi litere mari, astfel *edu*, *Edu*, sau *EDU* înseamnă acelaşi lucru. Componentele numelor pot avea o lungime de cel mult 63 caractere, iar întreaga cale de nume nu trebuie să depăşească 255 de caractere.

În principiu, domeniile pot fi inserate în arbore în două moduri diferite. De exemplu, *cs.yale.edu* ar putea la fel de bine să fie inclus în domeniul ţării *us* ca *csyale.ct.us.* În practică, totuşi, aproape toate organizaţiile din Statele Unite sunt repartizate după criteriul generic, iar aproape toate din afara Statelor Unite fac parte din domeniul ţării lor. Nu există nici o regulă împotriva înregistrării sub două domenii de nivel superior, însă puţine organizaţii în afară de cele multinaţionale o fac (de exemplu, *sony.com* şi *sony.nl*).

Fiecare domeniu controlează cum sunt alocate domeniile de sub el. De exemplu, Japonia are domeniile *ac.jp* şi *co.jp* echivalente cu *edu* şi *com.* Olanda nu face nici o distincţie şi pune toate organizaţiile direct sub *nl.* Astfel următoarele trei nume sunt toate departamente de calculatoare (computer science) din universităţi:

1. *cs.yale.edu* (Universitatea Yale din Statele Unite).
2. *cs.vn.nl* (Vrije Universiteit în Olanda).
3. *cs.keio.ac.jp* (Universitatea Keio din Japonia).

Pentru a crea un nou domeniu, se cere permisiunea domeniului în care va fi inclus. De exemplu, dacă un grup VLSI de la Yale doreşte să fie cunoscut ca *vlsi.csyale.edu,* acesta are nevoie de permisiunea celui care administrează *cs.yale.edu.* Similar, dacă este acreditată o nouă universitate, să zicem Universitatea din Northern South Dakota, ea trebuie să ceară administratorului domeniului *edu* să-i atribuie *unsd.edu.* În acest mod sunt evitate conflictele de nume şi fiecare domeniu poate ţine evidenţa tuturor subdomeniilor sale. Odată ce un nou domeniu a fost creat şi înregistrat, el poate crea sub- domenii, cum ar fi *cs.unsd.edu*, fără a cere permisiunea de la cineva din partea superioară a arborelui.

Atribuirea de nume respectă graniţele organizaţionale, nu pe cele ale reţelelor fizice. De exemplu, dacă departamentele de ştiinţa calculatoarelor şi de inginerie electrică sunt localizate în aceeaşi clădire şi folosesc aceeaşi reţea locală, ele pot avea totuşi domenii distincte. Similar, dacă departamentul de ştiinţa calculatoarelor este împărţit în două clădiri (Babbage Hall şi Turing Hall), toate sistemele gazdă din ambele clădiri aparţin aceluiaşi domeniu.

7.1.2 Înregistrări de resurse

Fiecărui domeniu, fie că este un singur calculator gazdă, fie un domeniu de nivel superior, îi poate fi asociată o mulţime de înregistrări de resurse (**resource records**). Pentru un singur sistem gazdă, cea mai obişnuită înregistrare de resursă este chiar adresa IP, dar există multe alte tipuri de înregistrări de resurse. Atunci când un resolver trimite un nume de domeniu către un DNS, ceea ce va primi ca

răspuns sunt înregistrările de resurse asociate acelui nume. Astfel, adevărata funcţie a DNS este să realizeze corespondenţa dintre numele de domenii şi înregistrările de resurse.

O înregistrare de resursă este un 5-tuplu. Cu toate că, din raţiuni de eficienţă, înregistrările de resurse sunt codificate binar, în majoritatea expunerilor ele sunt prezentate ca text ASCII, câte o înregistrare de resursă pe linie. Formatul pe care îl vom utiliza este următorul:

```
Nume_domeniu Timp_de_viaţă Clasă Tip Valoare
```

Nume_domeniu (domain_name) precizează domeniul căruia i se aplică această înregistrare. În mod normal există mai multe înregistrări pentru fiecare domeniu şi fiecare copie a bazei de date păstrează informaţii despre mai multe domenii. Acest câmp este utilizat ca cheie de căutare primară pentru a satisface cererile. Ordinea înregistrărilor în baza de date nu este semnificativă.

Câmpul *Timp_de_viaţă* (time_to_live) dă o indicaţie despre cât de stabilă este înregistrarea. Informaţia care este foarte stabilă are asigurată o valoare mare, cum ar fi 86400 (numărul de secunde dintr-o zi). Informaţiei instabile îi este atribuită o valoare mică, cum ar fi 60 (1 minut). Vom reveni la acest punct mai târziu, când vom discuta despre utilizarea memoriei ascunse.

Al treilea câmp dintr-o înregistrare de resursă este *Clasa* (class). Pentru informaţiile legate de Internet este tot timpul *IN*. Pentru alte informaţii pot fi folosite alte coduri, însă în practică acestea se întâlnesc rar.

Câmpul *Tip* (type) precizează tipul înregistrării. Cele mai importante tipuri sunt prezentate în fig. 7-2.

Tip	Semnificaţie	Valoare
SOA	Start autoritate	Parametrii pentru această zonă
A	Adresa IP a unui sistem	Întreg pe 32 de biţi
MX	Schimb de poştă	Prioritate, domeniu dispus să accepte poştă
NS	Server de Nume	Numele serverului pentru acest domeniu
CNAME	Nume canonic	Numele domeniului
PTR	Pointer	Pseudonim pentru adresa IP
HINFO	Descriere siste m gazd ă	Unitate centrală şi sistem de operare în ASCII
TXT	Text	Text ASCII neinterpretat

Fig. 7-2. Principalele tipuri de înregistrări de resurse DNS.

O înregistrare *SOA* furnizează numele sursei primare de informaţii despre zona serverului de nume (descrisă mai jos), adresa de e-mail a administratorului, un identificator unic şi diverşi indicatori şi contoare de timp.

Cel mai important tip de înregistrare este înregistrarea *A* (adresă). Ea păstrează adresa IP de 32 de biţi a unui sistem gazdă. Fiecare sistem gazdă Internet trebuie să aibă cel puţin o adresă IP, astfel încât alte maşini să poată comunica cu el. Unele sisteme gazdă au două sau mai multe conexiuni în reţea, caz în care vor avea câte o înregistrare de tip *A* pentru fiecare conexiune (şi astfel pentru fiecare adresă IP).

Următoarea ca importanţă este înregistrarea *MX.* Aceasta precizează numele sistemului gazdă pregătit să accepte poşta electronică pentru domeniul specificat. El este folosit deoarece nu toate maşinile sunt pregătite să accepte poşta electronică pentru domeniul specificat. Dacă cineva vrea să-i trimită un e-mail, de

exemplu, lui *bill@microsoft.com*, sistemul care trimite trebuie să găsească un server la *microsoft.com* dispus să accepte e-mail. Înregistrarea *MX* poate să furnizeze această informaţie.

Înregistrările *NS* specifică serverele de nume. De exemplu, fiecare bază de date DNS are în mod normal o înregistrare *NS* pentru fiecare domeniu de pe primul nivel, astfel încât, de exemplu, poşta electronică să poată fi trimisă în zone îndepărtate ale arborelui de nume. Vom reveni la acest aspect mai târziu.

Înregistrările *CNAME* permit crearea pseudonimelor. De exemplu, o persoană familiarizată cu atribuirea numelor în Internet, care doreşte să trimită un mesaj unei persoane al cărei nume de conectare la un sistem de calcul din departamentul de calculatoare de la M.I.T. este *paul* poate presupune că adresa *paul@cs.mit.edu* este corectă. De fapt această adresă nu este corectă, deoarece domeniul departamentului de calculatoare de la M.I.T. este *lcs.mit.edu*. Totuşi, ca un serviciu pentru cei care nu ştiu acest lucru, M.I.T. poate crea o intrare *CNAME,* pentru a dirija persoanele şi programele în direcţia corectă. O astfel de intrare poate fi:

```
cs.mit.edu 86400 IN CNAME lcs.mit.edu
```

Ca şi *CNAME*, *PTR* se referă la un alt nume. Totuşi, spre deosebire de *CNAME,* care este în realitate numai o macro-definiţie, *PTR* este un tip de date DNS a cărui interpretare depinde de contextul în care este utilizat. În practică este aproape întotdeauna utilizat pentru asocierea unui nume cu o adresă IP, pentru a permite căutarea adresei IP şi obţinerea numelui maşinii corespunzătoare. Acestea se numesc **căutări inverse** (**reverse lookups**).

Înregistrările *HINFO* permit aflarea tipului de maşină şi de sistem de operare cărora le corespunde domeniul. În sfârşit, înregistrările *TXT* permit domeniilor să se autoidentifice într-un mod arbitrar. Aceste două tipuri de înregistrări sunt introduse pentru uşurinţa utilizatorului. Nici una dintre ele nu este necesară, astfel încât programele nu pot conta pe obţinerea lor (şi probabil că dacă le obţin nu le pot trata).

În final ajungem la câmpul *Valoare.* Acest câmp poate fi un număr, un nume de domeniu sau un şir ASCII. Semantica depinde de tipul de înregistrare. O scurtă descriere a câmpurilor *Valoare* pentru fiecare dintre principalele tipuri de înregistrări este dată în fig. 7-2.

Un exemplu de informaţie ce se poate găsi în baza de date DNS a unui domeniu este prezentat în fig. 7-3. Această figură prezintă o parte (semi-ipotetică) a bazei de date pentru domeniul *cs.vu.nl* prezentat în fig. 7-1. Baza de date conţine şapte tipuri de înregistrări de resurse.

```
;Baza de date pentru cs.vu.nl
cs.vu.nl.       86400   IN  SOA        star boss (9527, 7200, 7200, 241920,
cs.vu.nl.       86400   IN  TXT        „Divisie Wiskunde en Informatica.”
cs.vu.nl.       86400   IN  TXT        „Vrije Universiteit Amsterdam.”
cs.vu.nl.       86400   IN  MX         1 zephyr.cs.vu.nl.
cs.vu.nl.       86400   IN  MX         2 top.cs.vu.nl.
flits.cs.vu.nl. 86400   IN  HINFO      Sun U nix
flits.cs.vu.nl. 86400   IN  A          130.37.16.112
flits.cs.vu.nl. 86400   IN  A          192.31.231.165
```

flits.cs.vu.nl.	86400	IN	MX	1 flits.cs.vu.nl.
flits.cs.vu.nl.	86400	IN	MX	2 zephyr.cs.vu.nl.
flits.cs.vu.nl.	86400	IN	MX	3 top.cs.vu.nl.
www.cs.vu.nl.	86400	IN	CNAME	star.cs.vu.nl.
ftp.cs.vu.nl.	86400	IN	CNAME	zephyr.cs.vu.nl.
rowboat		IN	A	130.37.56.201
		IN	MX	1 rowboat
		IN	MX	2 zephyr
		IN	HINFO	Sun Unix
little-sister		IN	A	130.37.62.23
		IN	HINFO	Mac MacOS
laserjet		IN	A	192.31.231.216
		IN	HINFO	„HP LaserJet IIISi” Proprietary

Fig. 7-3. O parte dintr-o posibilă bază de date DNS pentru *cs.vu.nl.*

Prima linie necomentată din fig. 7-3 dă câteva informaţii de bază despre domeniu, de care nu ne vom ocupa. Următoarele două linii furnizează informaţii textuale despre amplasarea domeniului. Urmează două intrări care specifică primul şi al doilea loc unde se încearcă să se livreze poşta electronică trimisă pentru *persoana@cs.vu.nl.* Mai întâi se încearcă trimiterea la maşina *zephyr.* Dacă aceasta eşuează, atunci trebuie să se încerce la *top*.

După o linie liberă, adăugată numai pentru claritate, urmează linii care spun că *flits* este o staţie de lucru Sun care lucrează sub UNIX şi se specifică ambele sale adrese IP. Urmează trei variante de tratare a poştei electronice trimise la *flits.cs.vu.nl.* Prima alegere este, în mod natural, chiar *flits,* iar dacă nu se reuşeşte, se încearcă la *zephyr* şi apoi la *top*. Urmează un pseudonim, *www.cs.vu.nl*, astfel ca această adresă să poată fi utilizată fără a specifica o anumită maşină. Crearea acestui pseudonim permite ca *cs.vu.nl* să schimbe serverul www fără invalidarea adresei folosite în mod curent pentru adresarea lui. Un argument similar este valabil pentru *ftp.cs.vu.nl.*

Următoarele patru linii conţin o înregistrare tipică pentru o staţie de lucru, în acest caz *rowboat. cs.vu.nl*. Informaţia furnizează adresa IP, destinaţia primară şi secundară pentru poşta electronică şi informaţii despre maşină. Urmează o intrare pentru un sistem non-UNIX care nu este capabil să primească poşta el însuşi, urmat de o intrare pentru o imprimantă laser conectată la Internet.

Ceea ce nu este arătat (şi nu există în acest fişier) sunt adresele IP utilizate pentru a căuta adresele domeniilor de pe primul nivel. Acestea sunt necesare pentru a căuta sistemele gazdă aflate la distanţă, dar, deoarece ele nu fac parte din domeniul *cs.vu.nl,* nu se găsesc în acest fişier. Ele sunt furnizate de serverele rădăcină ale căror adrese IP sunt prezentate în fişierul de configurare a sistemului şi sunt încărcate în memoria ascunsă DNS atunci când este pornit serverul DNS. Există cam o duzină de servere rădăcină în lume şi fiecare ştie adresele IP ale tuturor celorlalte servere de domenii de nivel superior. Astfel, dacă o maşină ştie adresa IP a cel puţin unuia din serverele rădăcină, el poate căuta orice nume DNS.

7.1.3 Servere de nume

Teoretic, un singur server de nume poate conţine întreaga bază de date DNS şi poate să răspundă tuturor cererilor. În practică, acest server poate fi atât de încărcat, încât să devină de neutilizat. În afară de aceasta, dacă se defectează, va fi afectat întregul Internet.

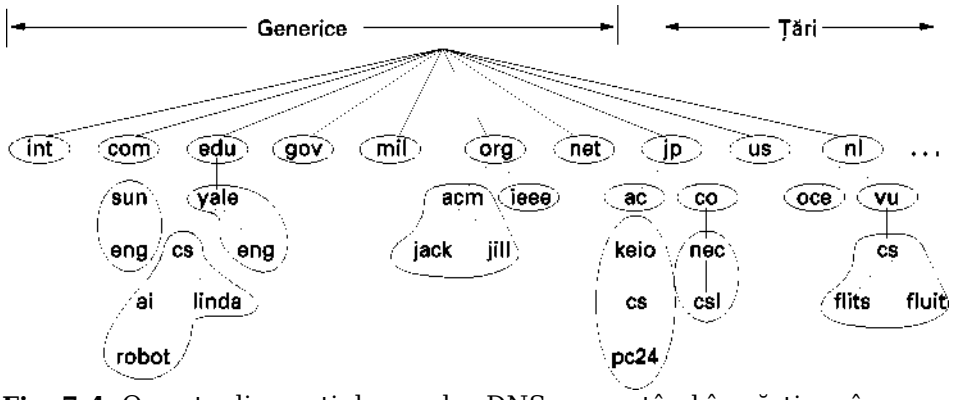


Fig. 7-4. O parte din spaţiul numelor DNS prezentând împărţirea în zone.

Pentru a evita problemele asociate cu existenţa unei singure surse de informaţie, spaţiul de nume DNS este împărţit în **zone** care nu se suprapun. O posibilă cale de împărţire a spaţiului de nume din fig. 7-l este arătată în fig. 7-4. Fiecare zonă conţine câte o parte a arborelui precum şi numele serve- re-lor care păstrează informaţia autorizată despre acea zonă. În mod normal, o zonă va avea un server de nume primar, care preia informaţia dintr-un fişier de pe discul propriu şi unul sau mai multe servere de nume secundare, care iau informaţiile de pe serverul primar. Pentru a îmbunătăţi fiabilitatea, unele servere pentru o zonă pot fi plasate în afara zonei.

Plasarea limitelor unei zone este la latitudinea administratorului ei. Această decizie este luată în mare parte bazându-se pe câte servere de nume sunt dorite şi unde să fie plasate. De exemplu, în fig. 7-4, Yale are un server pentru *yale.edu* care administrează *eng.yale.edu,* dar nu şi *cs.yale.edu,* care este o zonă separată cu propriile servere de nume. O astfel de decizie poate fi luată atunci când un de- partament ca cel de engleză nu doreşte să aibă propriul server de nume, în schimb departamentul de calculatoare doreşte. În consecinţă *cs.yale.edu* este o zonă separată, în timp ce zona *eng. yale.edu* nu este separată.

Atunci când un resolver are o cerere referitoare la un nume de domeniu, el transferă cererea unuia din serverele locale de nume. Dacă domeniul căutat este sub jurisdicţia serverului de nume, cum ar fi *ai.cs.yale.edu,* care este sub *cs.yale.edu,* el reîntoarce înregistrări de resurse autorizate. O **înregistrare autorizată** (**authoritative record**) este cea care vine de la autoritatea care adminis- trează înregistrarea şi astfel este întotdeauna corectă. Înregistrările autorizate se deosebesc de înregistrările din memoria ascunsă, care pot fi expirate.

Dacă, totuşi, domeniul se află la distanţă, iar local nu este disponibilă nici o

informaţie despre domeniul cerut, atunci serverul de nume trimite un mesaj de cerere la serverul de nume de pe primul nivel al domeniului solicitat. Pentru a clarifica acest proces să considerăm exemplul din fig. 7-5. Aici resolverul de pe *flits.cs.vu.nl* doreşte să ştie adresa IP a sistemului gazdă *linda.cs.yale.edu.* În pasul 1 trimite o cerere la serverul de nume local *cs.vu.nl.* Această cerere conţine numele de domeniu căutat, tipul *(A)* şi clasa *(IN).*

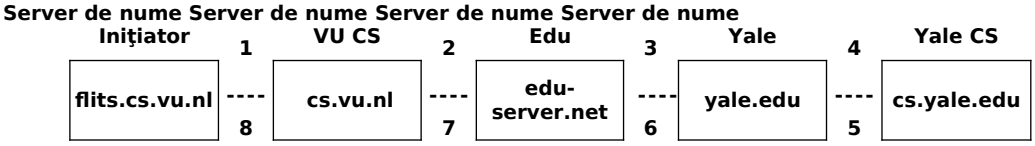


Fig. 7-5. Modul în care un resolver caută un nume aflat la distanţă, în opt paşi.

Să presupunem că serverul local de nume nu a avut niciodată o cerere pentru acest domeniu şi nu ştie nimic despre el. Poate solicita informaţii de la câteva servere de nume din apropriere, dar dacă nici unul dintre ele nu ştie, va trimite un pachet UDP la serverul pentru *edu* specificat în baza de date (vezi fig. 7-5), *edu-server.net.* Este puţin probabil ca acest server să cunoască adresa *linda.cs.yale.edu* şi probabil nu cunoaşte nici adresa *csyale.edu,* dar trebuie să cunoască adresele fiilor din subarbore, astfel că va transmite cererea la serverul de nume *yale.edu* (pas 3). Acesta va transmite cererea mai departe către *cs.yale.edu* (pas 4), care trebuie să aibă înregistrările autorizate de resurse. Deoarece fiecare cerere este de la un client la un server, înregistrarea de resursă cerută parcurge paşii 5 până la 8.

Odată ce aceste înregistrări de resurse ajung înapoi la serverul de nume *cs.vu.nl,* ele vor fi depuse în memoria ascunsă, pentru a fi folosite ulterior. Totuşi, această informaţie nu este autorizată, deoarece orice schimbare făcută la *cs.yale.edu* nu se va propaga spre toate serverele care au folosit-o. Din acest motiv intrările în memoria ascunsă nu ar trebui să aibă viaţă prea lungă. Acesta este motivul pentru care câmpul *Timp_de_viaţă* este inclus în fiecare înregistrare de resursă. El informează serverele de nume aflate la distanţă cât timp să menţină înregistrările în memoria ascunsă. Dacă o anumită maşină are de ani de zile aceeaşi adresă IP, această informaţie ar putea fi păstrată timp de o zi. Pentru informaţii mai volatile este mai sigur ca înregistrările să fie eliminate după câteva secunde sau un minut.

De menţionat că metoda de interogare descrisă aici este cunoscută ca metoda de **interogare re- cursivă** (**recursive query**), deoarece fiecare server care nu are informaţia cerută o caută în altă parte şi raportează. Este posibilă şi o altă variantă. În acest caz, atunci când o cerere nu poate fi rezolvată local, cererea eşuează, dar este întors numele următorului server de pe calea ce trebuie încercată. Unele servere nu implementează interogarea recursivă şi întorc întotdeauna numele următorului server la care să se încerce.

De asemenea merită menţionat faptul că atunci când un client DNS nu reuşeşte să primească un răspuns înainte de expirarea timpului de căutare, data viitoare va încerca un alt server. Se presupune că serverul este probabil nefuncţional, nu că cererea sau răspunsul s-au pierdut.

Deşi DNS este foarte important pentru funcţionarea corectă a Internetului, el nu face decât să pună în corespondenţă nume simbolice de maşini cu adresele lor IP. El nu ajută la localizarea oamenilor, resurselor, serviciilor sau obiectelor în general. Pentru localizarea acestora a fost definit un alt serviciu director, numit **LDAP** (**Lightweight Directory Access Protocol**, rom.: **Protocol de acces la cataloage simplificate**). El este o versiune simplificată a serviciului de cataloage OSI X.500, descris în RFC 2251. El organizează informaţia sub formă de arbore şi permite căutări pe diferite componente. Poate fi privit ca o carte de telefon obişnuită (de tipul „pagini albe”). Nu o să intrăm în amănunte referitoare la el în această carte, însă puteţi găsi mai multe informaţii în (Weltman şi Dahbura, 2000).

7.2 POŞTA ELECTRONICĂ

Poşta electronică, sau **e-mail**, cum este cunoscută de către numeroşii săi admiratori, există de peste două decenii. Înainte de 1990, era folosită în special în mediul academic. În timpul anilor 1990, a devenit cunoscută publicului larg şi a crescut exponenţial până la punctul în care numărul de mesaje electronice trimise pe zi este acum mult mai mare decât numărul de scrisori tradiţionale (adică pe hârtie).

E-mail-ul, ca majoritatea celorlalte forme de comunicare, are convenţiile şi stilurile sale proprii. În particular, el este foarte neprotocolar şi are un prag de folosire foarte scăzut. Oamenii care n-ar visa niciodată să sune la telefon sau chiar să scrie o scrisoare unei Persoane Foarte Importante nu ezită o secundă să trimită un e-mail neglijent.

E-mail-ul este plin de jargoane precum BTW (By The Way – apropo), ROTFL (Rolling On The Floor Laughing – a se tăvăli pe jos de râs) şi IMHO (În My Humble Opinion – după umila mea părere). Mulţi oameni folosesc în e-mail-urile lor câteva caractere ASCII numite **smileys** sau **emoticons** (faţa zâmbitoare şi faţa tristă). Câteva din cele mai interesante sunt reproduse în fig. 7-6. Pentru cei mai mulţi, rotirea cărţii cu 90 de grade în sensul acelor de ceasornic, le va face mai clare. Pentru o cărticică cu peste 650 smileys, vedeţi (Sanderson şi Dougherty, 1993).

Smiley	Semnificaţie	Smiley	Semnificaţie	Smiley	Semnificaţie
	Sunt fericit	=I:-)	Abe Lincoln	:+)	Nas mare
	Sunt trist, sup	=):-)	Unchiul Sam	:-))	Guşă
:-I	Sunt apatic	*<:-)	Moş Crăciun	:-{)	Mustaţă
;-)	Fac cu ochiu l	<:-(	Dunce	#:-)	Păr încurcat
:-(0)	Tip	(-:	Australian	8-)	Poartă ochelari
:-(*)	Vomit	:-)X	Om cu papion	C:-)	Cap mare

Fig. 7-6. Câteva smileys. Nu vor fi în examenul final :-)

Primele sisteme de poştă electronică constau pur şi simplu din protocoale de transfer de fişiere, cu convenţia ca prima linie a fiecărui mesaj (adică fişier) să conţină adresa receptorului. Cu timpul, limitările acestei abordări au devenit din ce în ce mai evidente. O parte dintre neajunsuri erau:

1. Trimiterea unui mesaj către un grup de persoane era incomodă. Managerii au nevoie adesea de această facilitate pentru a trimite note şi rapoarte tuturor subordonaţilor.
2. Mesajele nu aveau structură internă, făcând astfel dificilă prelucrarea lor cu ajutorul calculatorului. De exemplu, dacă un mesaj trimis mai departe era inclus în corpul altui mesaj, extragerea părţii incluse din mesajul primit era dificilă.
3. Iniţiatorul (transmiţătorul) nu ştia niciodată dacă mesajul a ajuns sau nu.
4. Dacă cineva avea în plan să plece în călătorie de afaceri pentru mai multe săptămâni şi dorea ca toată poşta primită în acest timp să fie preluată de către secretară, acest lucru nu era uşor de realizat.
5. Interfaţa cu utilizatorul era slab integrată cu sistemul de transmisie, cerând utilizatorilor ca întâi să editeze un fişier, apoi să părăsească editorul şi să apeleze programul de transfer de fişiere.
6. Nu era posibilă transmiterea de mesaje care să conţină o combinaţie de text, desene, facsimil şi voce.

Pe măsură ce s-a câştigat experienţă, au fost propuse sisteme de poştă electronică mai complicate. În 1982 au fost publicate propunerile cu privire la e-mail ale ARPANET, sub numele de RFC 821 (protocolul de transmisie) şi RFC 822 (formatul mesajelor). Revizii minore, RFC 2821 şi RFC 2822, au devenit standarde Internet, totuşi toată lumea se referă la e-mail gândindu-se la RFC 822.

În 1984, CCITT a emis recomandarea X.400. După două decenii de competiţie, sistemele de poştă electronică bazate pe RFC 822 sunt larg răspândite, în timp ce acelea bazate pe X.400 au dispărut. Modul în care un sistem încropit de o mână de absolvenţi de ştiinţa calculatoarelor a învins un standard internaţional oficial, puternic susţinut de către toate PTT-urile din lumea întreagă, de multe guverne şi de o parte substanţială a industriei calculatoarelor, ne aduce în minte povestea biblică a lui David şi Goliat.

Motivul succesului lui RFC822 nu este dat de faptul că ar fi atât de bun, ci acela că X.400 a fost atât de slab proiectat şi atât de complex, încât nimeni nu l-ar putea implementa bine. Având de ales între un sistem nesofisticat, dar care funcţionează, cum este cel bazat pe RFC822 şi sistemul de email X.400, presupus cu adevărat minunat, dar nefuncţional, majoritatea organizaţiilor l-au ales pe primul. Poate că este şi o lecţie în spatele acestei poveşti. De acea discuţia noastră referitoare la e-mail se va concentra asupra sistemului de e-mail din Internet.

7.2.1 Arhitectură şi servicii

În această secţiune vom furniza o prezentare de ansamblu a ceea ce pot face sistemele de poştă electronică şi cum sunt ele organizate. Aceste sisteme constau de obicei din două subsisteme: **agen- ţii-utilizator**, care permit utilizatorilor să citească şi să trimită scrisori prin poşta electronică şi **agenţii de transfer de mesaje**, care transportă mesajele de la sursă la destinaţie. Agenţii-utilizator sunt programe locale, care furnizează o metodă de a interacţiona cu sistemul de e-mail bazată pe comenzi, meniuri sau grafică. Agenţii de transfer de mesaje sunt, de

regulă, **demoni** de sistem, adică procese care se execută în fundal. Sarcina lor este să transfere mesajele prin sistem.

În general, sistemele de poştă electronică pun la dispoziţie cinci funcţii de bază. Să aruncăm o privire asupra lor.

Compunerea se referă la procesul de creare a mesajelor şi a răspunsurilor. Deşi pentru corpul mesajului poate fi folosit orice editor de texte, sistemul însuşi poate acorda asistenţă la adresare şi la completarea numeroaselor câmpuri antet ataşate fiecărui mesaj. De exemplu, când se răspunde la un mesaj, sistemul poate extrage adresa iniţiatorului din mesajul primit şi o poate insera automat în locul potrivit din cadrul răspunsului.

Transferul se referă la deplasarea mesajului de la autor la receptor. În mare, aceasta necesită stabilirea unei conexiuni la destinaţie, sau la o maşină intermediară, emiterea mesajului şi eliberarea conexiunii. Sistemul de poştă ar trebui să facă acest lucru singur, fără a deranja utilizatorul.

Raportarea se referă la informarea iniţiatorului despre ce s-a întâmplat cu mesajul. A fost livrat? A fost respins? A fost pierdut? Există numeroase aplicaţii în care confirmarea livrării este importantă şi poate avea chiar semnificaţie juridică. („Ştiţi, domnule judecător, sistemul meu de poştă electronică nu e foarte de încredere, aşa că presupun că citaţia electronică s-a pierdut pe undeva.”)

Afişarea mesajelor primite este necesară pentru ca utilizatorii să-şi poată citi poşta. Uneori sunt necesare conversii sau trebuie apelat un program de vizualizare special; de exemplu, dacă mesajul este un fişier PostScript, sau voce digitizată. Se mai încearcă uneori şi conversii simple şi formatări.

Dispoziţia este pasul final şi se referă la ceea ce face receptorul cu mesajul, după ce l-a primit. Posibilităţile includ eliminarea sa înainte de a-l citi, aruncarea sa după citire, salvarea sa ş.a.m.d. Ar trebui de asemenea să fie posibilă regăsirea şi recitirea de mesaje deja salvate, trimiterea lor mai departe, sau procesarea lor în alte moduri.

În plus faţă de aceste servicii de bază, unele sisteme de e-mail, în special cele interne companiilor, dispun de o gamă variată de facilităţi avansate. Să menţionăm pe scurt câteva dintre ele. Când utilizatorii se deplasează sau când sunt plecaţi pentru o perioadă de timp, pot dori ca poşta lor să fie trimisă acolo unde se găsesc, aşa că sistemul ar trebui să fie capabil să facă acest lucru automat.

Majoritatea sistemelor permit utilizatorilor să-şi creeze **cutii poştale (mailboxes)** pentru a păstra mesajele sosite. Sunt necesare comenzi de creare şi distrugere a cutiilor poştale, de inspectare a conţinutului acestora, de inserare şi de ştergere de mesaje din cutii poştale ş.a.m.d.

Managerii de companii au adesea nevoie să trimită un acelaşi mesaj fiecărui subordonat, client sau furnizor. Acest lucru dă naştere ideii de **listă de poştă (mailing list)**, care este o listă de adrese de poştă electronică. Când un mesaj este trimis la lista de poştă, copii identice ale sale sunt expediate fiecăruia dintre cei de pe listă.

Alte caracteristici evoluate sunt copii la indigo, poştă de prioritate mare, poştă secretă (criptată), receptori alternativi, dacă cel primar nu este disponibil, şi posibilitatea de a permite secretarelor să se ocupe de poşta primită de şefii lor.

Poşta electronică este în prezent folosită pe scară largă în industrie, pentru comunicaţie în cadrul companiilor. Aceasta permite unor angajaţi, răspândiţi la distanţe mari unii de ceilalţi, chiar şi peste mai multe fusuri orare, să coopereze la proiecte complexe. Eliminând majoritatea indiciilor cu privire la funcţie, vârstă şi gen, dezbaterile prin poşta electronică tind să se concentreze asupra ideilor şi nu a statutului din cadrul organizaţiei. Prin poşta electronică, o idee sclipitoare a unui student la cursurile de vară poate avea un impact mai mare decât una stupidă, venită de la un vicepreşedinte executiv.

O idee fundamentală în toate sistemele moderne de e-mail este distincţia dintre **plic** şi conţinutul său. Plicul încapsulează mesajul. Conţine toată informaţia necesară pentru transportul mesajului, cum ar fi destinaţia, adresa, prioritatea, nivelul de securitate, toate acestea fiind distincte de mesajul în sine. Agenţii de transfer de mesaje folosesc plicul pentru rutare (dirijare), aşa cum face şi oficiul poştal.

Mesajul din interiorul plicului conţine două părţi: **antetul** şi **corpul**. Antetul conţine informaţie de control pentru agenţii utilizator. Corpul mesajului se adresează în întregime utilizatorului uman. Plicurile şi mesajele sunt ilustrate în fig. 7-7.

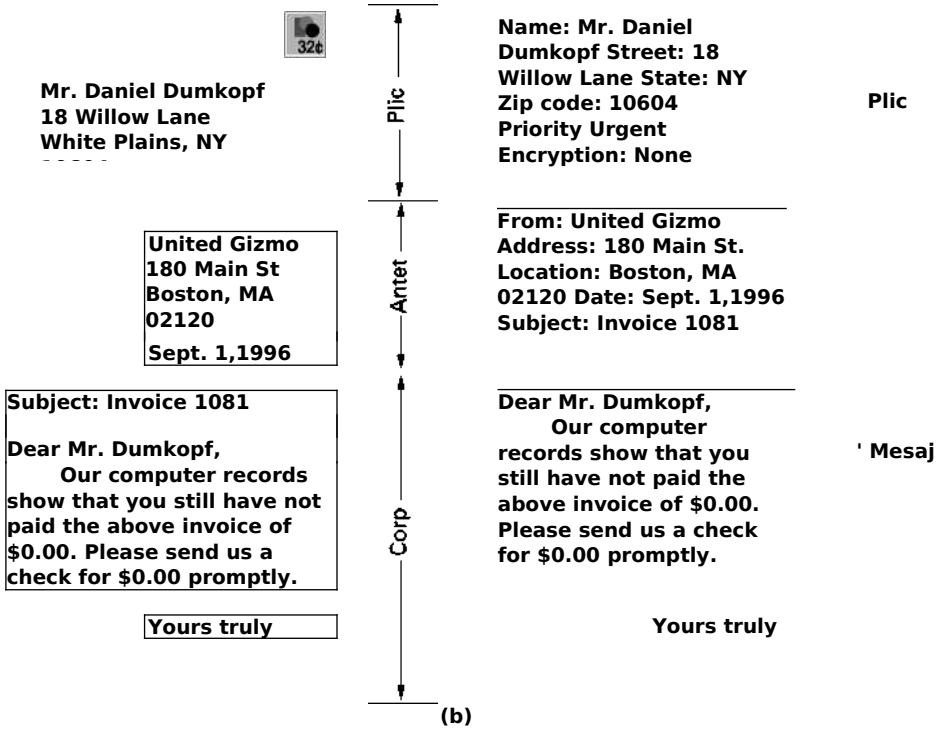


Fig. 7-7. Plicuri şi mesaje. (a) poştă clasică (b) poştă electronică.

7.2.2 Agentul utilizator

Sistemele de poştă electronică au, aşa cum am văzut, două părţi esenţiale: agenţii-utilizator şi agenţii de transfer de mesaje. În această secţiune ne vom uita la agenţii utilizator. Un agent utilizator este de obicei un program (uneori numit cititor de poştă) care acceptă o varietate de comenzi pentru compunerea, primirea şi răspunsul la mesaje, cât şi pentru manipularea cutiilor poştale. Unii agenţi-utilizator au o interfaţă sofisticată, dirijată prin meniuri sau icoane, care necesită un maus, în timp ce altele acceptă comenzi de câte un caracter, date de la tastatură. Funcţional însă, toţi aceştia sunt identici. Unele sisteme sunt dirijate prin meniuri sau icoane dau au şi alternative mai „scurte” pe tastatură.

Trimiterea poştei electronice

Pentru a trimite un mesaj prin poşta electronică, un utilizator trebuie să furnizeze mesajul, adresa destinaţie, şi eventual alţi câţiva parametri. Mesajul poate fi produs cu un editor de texte de sine- stătător, cu un program de procesare de text sau, eventual, cu un editor de texte specializat, construit în interiorul agentului utilizator. Adresa de destinaţie trebuie să fie într-un format cu care agentul utilizator să poată lucra. Mulţi agenţi-utilizator solicită adrese de forma *utilizator@adresă-dns.* Deoarece aceste lucruri au fost studiate anterior în acest capitol, nu vom relua materialul respectiv aici.

Oricum, merită notat că există şi alte forme de adresare. În particular, adresele X.400 arată radical diferit de cele DNS. Ele sunt compuse din perechi de forma *atribut = valoare,* separate de bare oblice. De exemplu:

/C=US/SP=MASSACHUSETTS/L=CAMBRIDGE/PA=360 MEMORIAL DR./CN=KEN SMITH/

Această adresă specifică o ţară, un stat, o localitate, o adresă personală şi un nume obişnuit (Ken Smith). Sunt posibile multe alte atribute, astfel încât poţi trimite mesaje cuiva al cărui nume nu-l ştii, atâta timp cât ştii suficiente alte atribute (de exemplu, compania şi funcţia). Cu toate că adresele X.400 sunt mult mai puţin convenabile decât cele DNS, cele mai multe sisteme de poştă electronică permit folosirea de **pseudonime** (**aliases**, uneori numite şi porecle) pentru obţinerea numelor sau adreselor de e-mail corecte ale unei persoane. În consecinţă, chiar şi cu adresele de tip X.400, de obicei nu este necesară scrierea în întregime a acelor şiruri ciudate.

Majoritatea sistemelor de e-mail acceptă liste de poştă, astfel că un utilizator poate trimite, cu o singură comandă, un acelaşi mesaj tuturor persoanelor dintr-o listă. Dacă lista de poştă este păstrată local, agentul-utilizator poate pur şi simplu să trimită câte un mesaj separat fiecăruia dintre receptorii doriţi. Dacă lista este păstrată la distanţă, atunci mesajele vor fi expandate acolo. De exemplu, dacă un grup de admiratori de păsări au o listă de poştă numită *birders,* instalată la *meadowlark. arizona.edu,* atunci orice mesaj trimis la *birders@meadowlark.arizona.edu* va fi dirijat către Universitatea din Arizona şi expandat acolo în mesaje individuale pentru toţi membrii listei de poştă, oriunde ar fi ei în lume. Utilizatorii acestei liste de poştă nu pot determina că aceasta este o listă de adrese. Ar putea fi la fel de bine cutia poştală personală a Prof. Gabriel O. Birders.

Citirea poştei electronice

În mod obişnuit, când este lansat un agent-utilizator, înainte de a afişa ceva pe ecran, el se va uita în cutia poştală a utilizatorului după mesajele care sosesc. Apoi poate anunţa numărul de mesaje din cutie, sau poate afişa pentru fiecare mesaj câte un rezumat de o linie, pentru ca apoi să aştepte o comandă.

Ca exemplu despre cum lucrează un agent-utilizator, să aruncăm o privire asupra unui scenariu tipic pentru poşta electronică. După lansarea agentului-utilizator, utilizatorul cere un rezumat al mesajelor sale. O imagine ca aceea din fig. 7-8 apare în acest caz pe ecran. Fiecare linie se referă la câte un mesaj. În acest exemplu, cutia poştală conţine opt mesaje.

#	Marcaje	Octeţi	Transmiţător	Subiect
1	K	103	Asw	Changes to MINIX
2	KA	634	Trudy	Not all Trudys are nasty
3	KF	451	Amy N. Wong	Request for information
4		123	Bal	Bioinformatics
5		103610	Kaashoek	Material on peer-to-peer
6		122	Frank	Re: Will you review a grant
7		311	Guido	Our paper has been accepted
8		120	Dmr	Re: My student's visit

Fig. 7-8. Un exemplu de afişare a conţinutului unei cutii poştale.

Fiecare linie de afişaj conţine câteva câmpuri extrase de pe plicul sau din antetul mesajului corespunzător. Într-un sistem simplu de poştă electronică, alegerea câmpurilor afişate este făcută în cadrul programului. Într-un sistem mai sofisticat, utilizatorul poate specifica ce câmpuri să fie afişate, furnizând un **profil al utilizatorului**, adică un fişier care descrie formatul de afişare. În exemplul considerat, primul câmp reprezintă numărul mesajului. Al doilea câmp, *Marcaje,* poate conţine un *K,* însemnând că mesajul nu este nou, dar a fost citit anterior şi păstrat în cutia poştală; un *A,* însemnând că deja s-a răspuns la acest mesaj; şi/sau un *F,* însemnând că mesajul a fost trimis mai departe altcuiva. Sunt de asemenea posibile şi alte marcaje.

Al treilea câmp specifică lungimea mesajului şi al patrulea spune cine a trimis mesajul. Din moment ce el este pur şi simplu extras din mesaj, acest câmp poate conţine prenume, nume complete, iniţiale, nume de cont, sau orice altceva şi-a ales transmiţătorul să pună. În sfârşit, câmpul *Subiect* specifică despre ce este mesajul, într-un scurt rezumat. Persoanele care omit să includă un câmp *Subiect* adesea descoperă că răspunsurile la scrisorile lor tind să nu obţină prioritate maximă.

După ce au fost afişate antetele, utilizatorul poate executa oricare dintre comenzile disponibile, ca de exemplu afişarea unui mesaj, ştergerea unui mesaj şi aşa mai departe. Sistemele mai vechi lucrau în mod text şi de obicei foloseau comenzi de un caracter pentru diversele operaţii, ca de exemplu T (scrie mesaj), A (răspunde la mesaj), D (şterge mesaj) şi F (trimite mai departe). Un argument specifica mesajul corespunzător. Sistemele mai recente folosesc interfeţe grafice. De obicei, utilizatorul selectează un mesaj cu mouse-ul şi apoi apasă pe o iconiţă pentru a scrie, răspunde la mesaj, sau pentru a-l trimite mai departe.

Poşta electronică a parcurs un drum lung de pe vremea când era doar transfer de fişiere. Agenţi- utilizator sofisticaţi fac posibilă manevrarea unui volum mare de scrisori. Pentru persoane care primesc şi trimit mii de mesaje pe an, asemenea instrumente sunt nepreţuite.

7.2.3 Formatele mesajelor

Să ne întoarcem acum de la interfaţa utilizator la formatul mesajelor de poştă electronică în sine. Mai întâi ne vom uita la e-mailul ASCII de bază, care utilizează RFC 822. După aceea ne vom concentra asupra extensiilor multimedia ale RFC 822.

RFC 822

Mesajele constau dintr-un plic simplu (descris în RFC 821), un număr de câmpuri antet, o linie goală şi apoi corpul mesajului. Fiecare câmp antet se compune (din punct de vedere logic) dintr-o singură linie de text ASCII, conţinând numele câmpului, două puncte, şi, pentru majoritatea câmpurilor, o valoare. RFC 822 a fost creat acum două decenii şi nu distinge clar plicul de câmpurile antet, cum ar face un standard nou. Cu toate că a fost corectat în RFC 2822, o refacere completă n-a fost posibilă datorită răspândirii sale largi. La o utilizare normală, agentul-utilizator construieşte un mesaj şi îl transmite agentului de transfer de mesaje, care apoi foloseşte unele dintre câmpurile antet pentru a construi plicul efectiv, o combinaţie oarecum demodată de mesaj şi plic.

Principalele câmpuri antet, legate de transportul de mesaje, sunt înfăţişate în fig. 7-9. Câmpul *To:* oferă adresa DNS a receptorului primar. Este permisă de asemenea existenţa de receptori multipli. Câmpul *Cc:* dă adresa oricărui receptor secundar. În termenii livrării, nu este nici o diferenţă între un receptor primar şi unul secundar. Este în întregime o deosebire psihologică, ce poate fi importantă pentru persoanele implicate, dar este neimportantă pentru sistemul de poştă. Termenul *Cc:* (Carbon copy – copie la indigo) este puţin depăşit, din moment ce calculatoarele nu folosesc indigo, dar este bine înrădăcinat. Câmpul *Bcc:* (Blind carbon copy – copie confidenţială la indigo) este la fel ca *Cc:*, cu excepţia că această linie este ştearsă din toate copiile trimise la receptorii primari şi secundari. Acest element permite utilizatorilor să trimită copii unei a treia categorii de receptori, fără ca cei primari şi secundari să ştie acest lucru.

Antet	Conţinut
To:	Adresa(ele) de e-mail a(le) receptorului(ilor) primar(i)
Cc:	Adresa(ele) de e-mail a(le) receptorului(ilor) secundar(i)
Bcc:	Adresa(ele) de e-mail pentru „blind carbon copy”
From:	Persoana sau persoanele care au creat mesajul
Sender:	Adresa de e-mail a transmiţătorului curent
Received:	Linie adăugată de fiecare agent de transfer de-a lungul
Return-Path:	Poate fi folosită pentru a identifica o cale de întoarcere la

Fig. 7-9. Câmpurile antet ale lui RFC 822, legate de transportul de mesaje.

Următoarele două câmpuri, *From:* şi *Sender:,* precizează cine a scris şi respectiv cine a trimis mesajul. Acestea pot să nu fie identice. De exemplu, se poate ca o directoare executivă să scrie un mesaj, dar ca secretara ei să fie cea care îl trimite efectiv. În acest caz, directoarea executivă va fi afişată în câmpul *From:* şi secretara în câmpul *Sender:.* Câmpul *From:* este obligatoriu, dar câmpul *Sender:* poate fi omis dacă este identic cu *From:*. Aceste câmpuri sunt necesare în cazul în care mesajul nu poate fi livrat şi trebuie returnat transmiţătorului.

O linie conţinând *Received:* este adăugată de fiecare agent de transfer de mesaje de pe traseu. Linia conţine identitatea agentului, data şi momentul de timp la care a fost primit mesajul şi alte informaţii care pot fi utilizate pentru găsirea defecţiunilor în sistemul de dirijare.

Câmpul *Return-Path:* este adăugat de agentul final de transfer de mesaje şi are în intenţie să indice cum se ajunge înapoi la transmiţător. În teorie, această informaţie poate fi adunată din toate antetele *Received:* (cu excepţia numelui cutiei poştale a transmiţătorului), dar rareori este completată aşa şi de obicei conţine chiar adresa transmiţătorului.

Antet	Conţinut
Date:	Data şi momentul de timp la care a fost trimis mesajul
Reply-To:	Adresa de e-mail la care ar trebui trimise răspunsurile
Message-Id:	Număr unic, utilizat ulterior ca referinţă pentru acest mesaj
În-Reply-To:	Identificatorul mesajului al cărui răspuns este mesajul curent
References:	Alţi identificatori de mesaje relevanţi
Keywords:	Cuvinte cheie alese de utilizator
Subject:	Scurt cuprins al mesajului, afişabil pe o singură linie

Fig. 7-10. Câteva câmpuri utilizate în antetul lui RFC 822.

În plus faţă de câmpurile din fig. 7-9, mesajele RFC 822 pot conţine de asemenea o varietate de câmpuri antet, folosite de agenţii-utilizator sau de receptorii umani. Cele mai des întâlnite dintre ele sunt prezentate în fig. 7-10. Majoritatea lor se explică de la sine, deci nu vom intra în detaliu la toate.

Câmpul *Reply-To:* este uneori utilizat când nici persoana care a compus mesajul, nici cea care l-a trimis nu vrea să vadă răspunsul. De exemplu, un director de marketing scrie un mesaj prin e-mail pentru a spune clienţilor despre un nou produs. Mesajul este trimis de o secretară, dar câmpul *Reply- To:* conţine şeful departamentului de vânzări, care poate răspunde la întrebări şi primi comenzi. Acest câmp este foarte folositor când transmiţătorul are două conturi de e-mail şi vrea ca răspunsul să ajungă în celălalt.

Documentul RFC 822 afirmă explicit că utilizatorilor le este permis să inventeze noi antete, atâta timp cât acestea încep cu şirul de caractere X-. Se garantează că nici o extindere ulterioară nu va folosi nume ce încep cu X-, pentru a evita conflictele (suprapunerile) dintre antetele oficiale şi cele personale. Uneori studenţii care fac pe deştepţii includ câmpuri de tipul *X-Fruit-of-the-Day:* sau *X- Disease-of-the-Week:,* care sunt legale, deşi nu întotdeauna clarificatoare.

După antete urmează corpul mesajului. Aici utilizatorii pot pune orice vor. Unii oameni îşi încheie mesajele cu semnături elaborate, incluzând caricaturi ASCII simple, citate din personalităţi mai mari sau mai mici, declaraţii politice şi declinări de tot felul (de exemplu: Corporaţia XYZ nu este răspunzătoare pentru părerile mele; de fapt nu poate nici măcar să le înţeleagă).

MIME – Multipurpose Internet Mail Extensions (extensii de poştă cu scop multiplu)

La începuturile ARPANET, poşta electronică consta exclusiv din mesaje de tip text, scrise în engleză şi exprimate în ASCII. Pentru acest context, RFC 822 realiza sarcina complet: specifică antetele, dar lăsa conţinutul în întregime în seama utilizatorilor. În zilele noastre, această abordare nu mai este adecvată pentru Internetul care se întinde în lumea întreagă. Problemele includ transmisia şi recepţia de:

1. Mesaje în limbi cu accente (de exemplu franceza şi germana).
2. Mesaje în alfabete ne-latine (de exemplu ebraică şi rusă).
3. Mesaje în limbi fără alfabet (de exemplu chineză şi japoneză).
4. Mesaje care nu conţin text deloc (de exemplu audio şi video).

O soluţie posibilă a fost propusă în RFC 1341 şi actualizată în RFC-urile 2045-2049. Această soluţie, numită **MIME (Multipurpose Internet Mail Extensions),** este în prezent larg utilizată. O vom descrie în continuare. Pentru informaţii suplimentare în legătură cu MIME, vedeţi RFC-urile.

Ideea fundamentală a MIME este să continue să folosească formatul RFC 822, dar să adauge structură corpului mesajului şi să definească reguli de codificare pentru mesajele non-ASCII. Deoarece respectă RFC 822, mesajele MIME pot fi trimise utilizând programele şi protocoalele de poştă existente. Tot ceea ce trebuie modificat sunt programele de transmitere şi recepţie, pe care utilizatorii le pot face ei înşişi.

Antet	Conţinut
MIME-Version:	Identifică versiunea de MIME
Content-Description:	Şir adresat utilizatorului care spune ce este în
Content-Id:	Identificator unic
Content-Transfer-	Cum este împachetat corpul pentru
Content-Type:	Natura mesajului

Fig. 7-11. Antetele RFC 822 adăugate de către MIME.

MIME defineşte cinci noi antete de mesaje, aşa cum se arată în fig. 7-11. Primul dintre acestea specifică pur şi simplu agentului-utilizator care primeşte mesajul că este vorba de un mesaj MIME şi ce versiune de MIME utilizează. Orice mesaj care nu conţine un antet *MIME-Version:* este presupus ca fiind un mesaj în text pur, în engleză, şi este procesat ca atare.

Antetul *Content-Description:* este un şir de caractere ASCII specificând ce este în mesaj. Acest antet este necesar pentru ca receptorul să ştie dacă merită să decodifice şi să citească mesajul. Dacă şirul de caractere spune: " Fotografia hamsterului Barbarei" şi persoana care primeşte mesajul nu este un mare iubitor de hamsteri, mesajul va fi probabil mai curând aruncat, decât decodificat într-o fotografie color de înaltă rezoluţie.

Antetul *Content-Id:* identifică conţinutul. Utilizează acelaşi format ca antetul standard *Message-Id:*.

Antetul *Content-Transfer-Encoding:* arată cum este împachetat pentru transmisie corpul mesajului, într-o reţea care poate ridica obiecţii la majoritatea caracterelor diferite de litere, cifre şi semne de punctuaţie. Sunt furnizate cinci scheme (plus o evadare către noi scheme). Cea mai simplă schemă se referă chiar la text ASCII. Caracterele ASCII utilizează 7 biţi şi pot fi transportate direct prin protocolul de e-mail, atâta timp cât nici o linie nu are mai mult de 1000 de caractere.

Următoarea schemă ca simplitate este cam acelaşi lucru, dar utilizează caractere de câte 8 biţi, reprezentând toate valorile de la 0 la 255 inclusiv. Această schemă de codificare încalcă protocolul (original) de e-mail utilizat în Internet, dar este folosită de unele părţi ale Internetului, care implementează nişte extensii ale protocolului original. În timp ce declararea codificării nu o face să devină legală, faptul că o avem explicit poate cel puţin să lămurească lucrurile atunci când ceva merge prost. Mesajele utilizând codificarea de 8 biţi trebuie încă să respecte lungimea maximă a liniei, care este standard.

Este chiar mai rău în cazul mesajelor care utilizează codificare binară. Aceste mesaje sunt fişiere binare arbitrare, care nu numai că utilizează toţi cei 8 biţi, dar nu respectă nici limita de linie de 1000 de caractere. Programele executabile intră în această categorie. Nu se acordă nici o garanţie că mesaje binare vor ajunge corect, dar mulţi le trimit oricum.

Modalitatea corectă de a codifica mesaje binare este de a utiliza **codificarea în bază 64**, numită uneori **armură ASCII.** În această schemă, grupuri de câte 24 de biţi sunt împărţite în patru unităţi de câte 6 biţi, fiecare dintre aceste unităţi fiind transmisă ca un caracter ASCII legal. Codificarea este „A" pentru 0, „B" pentru 1, ş.a.m.d., urmate de cele 26 de litere mici, cele 10 cifre, şi în cele din urmă + şi / pentru 62 şi respectiv 63. Secvenţele == şi = sunt utilizate pentru a arăta că ultimul

grup a conţinut doar 8 sau respectiv 16 biţi. Se ignoră secvenţele carriage return şi line feed, astfel că ele pot fi inserate după dorinţă, pentru a păstra liniile suficient de scurte. Utilizând această schemă pot fi trimise sigur texte binare arbitrare.

Pentru mesajele care sunt aproape în întregime ASCII şi conţin puţine caractere ne-ASCII, codificarea în bază 64 este oarecum ineficientă. În locul acesteia se utilizează o codificare numită **quoted- printable-encoding** (codificare afişabilă marcată). Aceasta este o codificare de tip ASCII pe 7 biţi, având toate caracterele cu cod mai mare de 127 codificate sub forma unui semn egal urmat de valoarea caracterului reprezentată prin două cifre hexazecimale.

Rezumând, datele binare ar trebui trimise codificate în bază 64 sau sub formă quoted-printable. Când există motive întemeiate pentru a nu utiliza una dintre aceste scheme, este posibil să se specifice în antetul Content-Transfer-Encoding: o codificare definită de către utilizator.

Ultimul antet înfăţişat în fig. 7-11 este cu adevărat cel mai interesant. El specifică natura corpului mesajului. În RFC 2045 sunt definite şapte tipuri, fiecare având unul sau mai multe subtipuri. Tipul şi subtipul sunt separate printr-o bară oblică (slash), ca în:

```
Content-Type: video/mpeg
```

Subtipul trebuie precizat explicit în antet; nu sunt furnizate valori implicite. Lista iniţială de tipuri şi subtipuri specificate în RFC 2045 este prezentată în fig. 7-12. De atunci au fost adăugate multe altele, introducându-se intrări adiţionale de fiecare dată când a devenit necesar.

Să parcurgem acum lista tipurilor. Tipul *text* este utilizat pentru text simplu. Combinaţia *text/plain* este folosită pentru mesaje obişnuite care pot fi afişate de îndată ce sunt primite, fără codificare sau procesare ulterioară. Această opţiune permite ca mesajele obişnuite să fie transportate în MIME adăugând doar câteva antete suplimentare.

Tip	Subtip	Descriere
Text	Plain	Text neformatat
	Enriched	Text incluzând comenzi simple de formatare
Image	Gif	Imagini fixe în format GIF
	Jpeg	Imagini fixe în format JPEG
Audio	Basic	Sunet
Video	Mpeg	Film în format MPEG
Application	Octet-stream	Secvenţă neinterpretată de octeţi
	Postscript	Un document afişabil în PostScript
Message	Rfc822	Un mesaj MIME RFC 822
	Partial	Mesajul a fost fragmentat pentru transmisie
	External-body	Mesajul în sine trebuie adus din reţea
Multipart	Mixed	Părţi independente în ordine specificată
	Alternative	Acelaşi mesaj în formate diferite
	Parallel	Părţile trebuie vizualizate simultan
	Digest	Fiecare parte este un mesaj RFC 822 complet

Fig. 7-12. Tipurile şi subtipurile aparţinând MIME definite în RFC 2045.

Subtipul *text/enriched* permite includerea în text a unui limbaj simplu de marcare. Acest limbaj furnizează o modalitate independentă de sistem pentru a

exprima scrierea cu caractere aldine sau cursive, dimensiunile, alinierea, distanţele dintre rânduri, folosirea de indici superiori sau inferiori şi paginarea simplă. Limbajul de marcare se bazează pe SGML, Standard Generalized Markup Language (limbajul standard generalizat de marcare), folosit de asemenea ca bază pentru HTML, utilizat în World Wide Web. De exemplu mesajul

```
The <bold> time </bold> has come the <italic> walrus </italic> said ...
```

ar fi afişat sub forma:

The **time** has come the *walrus* said...

Depinde de sistemul receptor să aleagă interpretarea potrivită. Dacă sunt disponibile caractere aldine şi cursive, acestea vor putea fi folosite; altfel, pentru a scoate în evidenţă se pot utiliza culori, scriere cu clipire sau video-invers etc. Sisteme diferite pot face alegeri diferite.

Când Web-ul a devenit popular, a fost adăugat un nou subtip, *text/html* (în RFC 2854) pentru a permite paginilor Web să fie trimise într-un e-mail de tip RFC 822. Un subtip pentru sistemul extins de marcare, *text/xml,* este definit în RFC 3023. Vom studia HTML şi XML mai târziu în acest capitol.

Următorul tip MIME este *image*, utilizat pentru trimiterea de imagini fixe. În zilele noastre sunt utilizate multe formate, atât cu, cât şi fără compresie, pentru a păstra şi transmite imagini. Două dintre acestea, GIF şi JPEG, sunt recunoscute de aproape toate programele de navigare, dar există şi altele care au fost adăugate la lista originală.

Tipurile *video* şi *audio* sunt pentru imagini în mişcare şi respectiv pentru imagini cărora li se asociază şi sunet. Trebuie notat că *video* include doar informaţia video, nu şi coloana sonoră. Dacă trebuie transmis un film cu sunet, s-ar putea ca porţiunile audio şi video să trebuiască să fie transmise separat, depinzând de sistemul de codificare utilizat. Primul format video definit a fost cel inventat de cei ce se intitulează modest Moving Picture Experts Group (MPEG – Grupul de experţi în imagini în mişcare), dar de atunci au fost adăugate şi altele. În plus faţă de *audio/basic,* un nou tip audio, *audio/mpeg* a fost adăugat în RFC 3003 pentru a permite oamenilor să trimită fişiere MP3 prin e-mail.

Tipul *application* este utilizat ca un colector pentru formatele care necesită prelucrare externă, neidentificate de nici unul dintre celelalte tipuri. Un *octet-stream* este doar o secvenţă de octeţi neinterpretaţi. La primirea unui asemenea flux, un agent-utilizator ar trebui probabil să-l afişeze, suge- rându-i utilizatorului să-l copieze într-un fişier şi cerându-i un nume pentru acesta. Procesarea ulterioară este apoi la latitudinea utilizatorului.

Celălalt subtip definit este *postscript*, care se referă la limbajul PostScript, produs de Adobe Systems şi larg utilizat pentru descrierea paginilor imprimate. Multe imprimante au înglobate interpretoare PostScript. Deşi un agent-utilizator poate pur şi simplu să apeleze un interpretor PostScript extern pentru a interpreta fişierele PostScript primite, acest lucru nu este lipsit de pericole. PostScript este un întreg limbaj de programare. Dându-i-se destul timp, o persoană suficient de masochistă ar putea scrie în PostScript un compilator de C, sau un sistem de management de baze de date. Afişarea unui mesaj primit în format PostScript se

face executând programul PostScript conţinut de acesta. Pe lângă afişarea unui text, acest program poate citi, modifica, sau şterge fişierele utilizatorului şi poate avea şi alte efecte laterale neplăcute.

Tipul *message* permite încapsularea în întregime a unui mesaj în altul. Această schemă este utilă, de exemplu pentru trimiterea mai departe a e-mailului, cu *forward.* Când un mesaj RFC 822 complet este încapsulat într-un mesaj exterior, ar trebui utilizat subtipul *rfc822.*

Subtipul*partial* face posibilă împărţirea unui mesaj încapsulat în bucăţi de mesaj şi trimiterea separată a acestora (de exemplu, dacă mesajul încapsulat este prea lung). Parametrii fac posibilă re- asamblarea în ordinea corectă a tuturor părţilor, la destinaţie.

Şi în sfârşit, subtipul *external-body* pate fi utilizat pentru mesaje foarte lungi (de exemplu, filme video). În loc de a include fişierul MPEG în mesaj, se dă o adresă FTP şi agentul utilizator al receptorului o poate aduce din reţea în momentul în care este necesar. Această facilitate este în special utilă când se trimite un film la o întreagă listă de poştă şi se presupune că doar câţiva dintre membrii acesteia îl vor vedea (gândiţi-vă la e-mailurile inutile, conţinând reclame video).

From: elinor@abcd.com
To: carolyn@xyz.com
MIME-Version: 1.0
Message-Id: <0704760941.AA00747@abcd.com>
Content-Type: multipart/alternative; boundary=qwertyuiopasdfghjklzxcvbnm Subject: Pământul înconjoară soarele de un număr întreg de ori

Acesta este preambulul. Agentul utilizator îl ignoră. O zi bună.

--qwertyuiopasdfghjklzxcvbnm Content-Type: text/richtext

Happy birthday to you Happy birthday to you Happy birthday dear <bold> Carolyn </bold> Happy birthday to you

--qwertyuiopasdfghjklzxcvbnm
Content-Type: message/external-body; access-type=„anon-ftp”;
site=„bicycle.abcd.com”; directory=„pub”;
name=„birthday.snd;

content-type: audio/basic content-transfer-encoding: base64 --qwertyuiopasdfghjklzxcvbnm

Fig. 7-13. Un mesaj multipart conţinând alternative de tip text formatat şi audio.

Ultimul tip este *multipart,* care permite unui mesaj să conţină mai multe părţi, începutul şi sfârşitul fiecărei părţi fiind clar delimitat. Subtipul *mixed* permite fiecărei părţi să fie diferită de celelalte, fără a avea o structură adiţională impusă. Multe programe de e-mail permit utilizatorului să aibă una sau mai multe părţi ataşate la un mesaj text. Acestea sunt trimise folosind tipul *multipart.*

În contrast cu tipul *multipart,* subtipul *alternative* permite ca fiecare parte să conţină acelaşi mesaj, dar exprimat într-un alt mediu sau într-o codificare diferită. De exemplu, un mesaj ar putea fi trimis ca ASCII simplu, ca text formatat şi ca PostScript. Un agent-utilizator proiectat corespunzător, la primirea unui asemenea mesaj, îl va afişa, dacă va fi posibil, în PostScript. A doua alegere va fi textul formatat. Dacă nici una dintre aceste alternative nu ar fi posibilă, s-ar afişa text

ASCII obişnuit. Părţile ar trebui ordonate de la cea mai simplă, la cea mai complexă, pentru a ajuta receptorii care folosesc agenţi-utilizator pre-MIME să înţeleagă mesajul (chiar şi un utilizator pre-MIME poate citi text ASCII simplu). Subtipul *alternative* poate fi folosit de asemenea pentru limbaje multiple. În acest context, Rosetta Stone poate fi privit ca precursor al mesajului de tip *multipart/alternative.*

Un exemplu multimedia este prezentat în fig. 7-13. Aici, o felicitare este transmisă atât sub formă de text cât şi sub formă de cântec. Dacă receptorul are facilităţi audio, agentul utilizator va aduce fişierul de sunet, *birthday.snd* şi îl va interpreta. Dacă nu, versurile vor fi afişate pe ecran într-o linişte de mormânt. Părţile sunt delimitate de două cratime urmate de şirul (definit de utilizator) specificat în parametrul *boundary.*

Observaţi că antetul *Content-Type* apare în trei poziţii în acest exemplu. La primul nivel indică faptul că mesajul are mai multe părţi. În cadrul fiecărei părţi specifică tipul şi subtipul acesteia. În sfârşit, în corpul celei de-a doua părţi, este necesar pentru a indica agentului utilizator ce fel de fişier extern trebuie să aducă. Pentru a exprima uşoara diferenţă de utilizare, s-au folosit litere mici, deşi toate antetele sunt *case insensitive* (nu fac diferenţă între literele mari şi cele mici). Antetul *contenttransfer-encoding* este în mod similar necesar pentru orice corp extern care nu este codificat ca ASCII pe 7 biţi.

Întorcându-ne la subtipurile corespunzătoare mesajelor *multipart*, vom spune că mai există două posibilităţi. Subtipul *parallel* este utilizat când toate părţile trebuie să fie interpretate simultan. De exemplu, adesea filmele au un canal audio şi unul video. Ele sunt mai de efect dacă aceste două canale sunt interpretate în paralel şi nu consecutiv.

În sfârşit, subtipul *digest* este utilizat când multe mesaje sunt împachetate împreună, într-unul compus. De exemplu, nişte grupuri de dialog de pe Internet pot aduna mesaje de la abonaţii lor şi apoi să le trimită în afară ca un singur mesaj de tip *multipart/digest.*

7.2.4 Transferul mesajelor

Sistemul de transfer de mesaje se ocupă cu transmiterea mesajelor de la expeditor la receptor. Cea mai simplă cale de a realiza acest lucru constă în stabilirea unei conexiuni de transport de la maşina sursă la cea de destinaţie şi apoi, pur şi simplu în trimiterea mesajului. După ce examinăm cum se face acest lucru în mod normal, vom studia câteva situaţii în care metoda nu funcţionează şi vom vedea ce trebuie făcut în aceste cazuri.

SMTP – Simple Mail Transfer Protocol (Protocol simplu de transfer de poştă)

În cadrul Internetului poşta electronică este livrată prin stabilirea de către maşina sursă a unei conexiuni TCP la portul 25 al maşinii de destinaţie. La acest port se află un demon de e-mail care ştie **SMTP (Simple Mail Transfer Protocol)**. Acest demon acceptă conexiunile şi copiază mesajele de la ele în cutiile poştale corespunzătoare. Dacă mesajul nu poate fi livrat, se returnează transmiţă- torului un raport de eroare conţinând prima parte a mesajului nelivrat.

SMTP este un protocol simplu de tip ASCII. După stabilirea conexiunii TCP la portul 25, maşina transmiţătoare, operând în calitate de client, aşteaptă ca maşina receptoare, operând ca server, să vorbească prima. Serverul începe prin a trimite o linie de text, declarându-şi identitatea şi spunând dacă este pregătit sau nu să primească mesaje. Dacă nu este, clienţii eliberează conexiunea şi încearcă din nou mai târziu.

```
                    S: 220 xyz.com SMTP service ready
C: HELO abcd.com
                    S: 250 xyz.com says hello to abcd.com
C: MAIL FROM: <elinor@abcd.com>
                    S: 250 sender ok
C: RCPT TO: <carolyn@xyz.com>
                    S: 250 recipient ok
C: DATA
          S: 354 Trimite mail; terminat cu "." pe linie nouă
C: From: elinor@abcd.com
C: To: carolyn@xyz.com
C: MIME-Version: 1.0
C: Message-Id: <0704760941.AA00747@abcd.com>
C: Content-Type: multipart/alternative; boundary=qwertyuiopasdfghjklzxcvbnm
C: Subject: Pământul înconjoară soarele de un număr întreg de ori
C:
C: Acesta este preambulul. Agentul utilizator îl ignoră. O zi bună.
C:
C: --qwertyuiopasdfghjklzxcvbnm
C: Content-Type: text/enriched
C:
C: Happy birthday to you
C: Happy birthday to you
C: Happy birthday dear <bold> Carolyn </bold>
C: Happy birthday to you
C:
C: --qwertyuiopasdfghjklzxcvbnm
C: Content-Type: message/external-body;
C: access-type="anon-ftp";
C: site="bicycle.abcd.com";
C: directory="pub";
C: name="birthday.snd";
C:
C: content-type: audio/basic
C: content-transfer-encoding: base64
C: --qwertyuiopasdfghjklzxcvbnm
C: .
                    S: 250 message accepted
C:QUIT
                    S: 221 xyz.com closing connection
```

Fig. 7-14. Transferul unui mesaj de la *elinor@abcd.com* la *carolyn@zxy.com*.

Dacă serverul este dispus să primească e-mail, clientul anunţă de la cine vine scrisoarea şi cui îi este adresată. Dacă un asemenea receptor există la destinaţie, serverul îi acordă clientului permisiunea să trimită mesajul. Apoi clientul trimite mesajul şi serverul îl confirmă. În general nu este necesară ataşarea unei sume de control deoarece TCP furnizează un flux sigur de octeţi. Dacă mai există şi alte

mesaje, acestea sunt trimise tot acum. Când schimbul de mesaje, în ambele direcţii, s-a încheiat, conexiunea este eliberată. În fig. 7-14 este prezentată o mostră de dialog referitoare la trimiterea mesajului din fig. 7-13, incluzând codurile numerice utilizate de SMTP. Liniile trimise de client sunt marcate cu C:, iar cele trimise de server cu S:.

Câteva comentarii în legătură cu fig. 7-14 ar putea fi utile. Prima comandă a clientului este într- adevăr *HELO.* Din posibilele abrevieri de patru caractere ale cuvântului *HELLO,* aceasta are numeroase avantaje faţă de concurenta sa cea mai mare. Motivul pentru care toate comenzile trebuiau să aibă patru caractere s-a pierdut în negura vremii.

În Fig.7-14, mesajul este trimis la un singur receptor şi de aceea este utilizată o singură comandă *RCPT.* Mai multe asemenea comenzi sunt permise pentru a trimite un singur mesaj mai multor receptori. Fiecare dintre ele este confirmată sau rejectată individual. Chiar dacă unii dintre receptori sunt rejectaţi (deoarece ei nu există la destinaţie), mesajul poate fi trimis celor rămaşi.

În sfârşit, deşi sintaxa comenzilor de patru caractere de la client este rigid specificată, sintaxa replicilor este mai puţin rigidă. Doar codul numeric contează cu adevărat. Fiecare implementare poate pune după cod ce şiruri de caractere vrea.

Pentru a înţelege mai bine cum funcţionează SMTP şi câteva din celelalte protocoale descrise în acest capitol, încercaţi-le! În orice caz, mai întâi mergeţi la o maşină conectată la Internet. Într-un sistem UNIX introduceţi comanda:

```
telnet mail.isp.com 25
```

înlocuind numele DNS cu numele serverului de mail al ISP-ului dvs. Pe un sistem Windows, faceţi clic pe Start, apoi Run, apoi tastaţi comanda în căsuţa de dialog. Această comandă va stabili o conexiune Telnet (adică TCP) pe portul 25 pe maşina respectivă. Portul 25 este portul SMTP (vezi Fig. 627 pentru câteva porturi uzuale). Probabil o sa primiţi un răspuns de genul:

```
Trying 192.30.200.66...
Connected to mail.isp.com
Escape character is '^A]'.
220 mail.isp.com Smail #74 ready at Thu, 25 Sept 2002 13:26 +0200
```

Primele trei linii spun ce face telnet-ul. Ultima linie este de la serverul SMTP de pe maşina de la distanţă, anunţând disponibilitatea acesteia de a vorbi cu dvs. şi de a accepta e-mail. Pentru a vedea ce comenzi acceptă, tastaţi:

```
HELP
```

De aici înainte, este posibilă o secvenţă de comenzi ca cea din fig. 7-14, începând cu comanda *HELO* dată de client.

E bine de notat faptul că folosirea liniilor de text ASCII pentru comenzi nu e un accident. Cele mai multe protocoale Internet funcţionează aşa. Folosirea textului ASCII face ca protocoalele foarte uşor de testat şi depanat. Ele pot fi testate trimiţând manual comenzi, cum am văzut mai sus, pentru care copiile mesajelor (eng.: dumps) sunt uşor de citit.

Chiar dacă protocolul SMTP este bine definit, mai pot apărea câteva probleme. O

problemă este legată de lungimea mesajelor. Unele implementări mai vechi nu pot să lucreze cu mesaje mai mari de 64KB. O altă problemă se referă la expirări de timp (timeout). Dacă acestea diferă pentru server şi client, unul din ei poate renunţa, în timp ce celălalt este încă ocupat, întrerupând conexiunea în mod neaşteptat. În sfârşit, în unele situaţii, pot fi lansate schimburi infinite de mesaje. De exemplu, dacă maşina 1 păstrează lista de poştă A şi maşina 2 lista de poştă B şi fiecare listă conţine o intrare pentru cealaltă, atunci orice mesaj trimis oricăreia dintre cele două liste va genera o cantitate nesfârşită de trafic de e-mail.

Pentru a atinge câteva dintre aceste probleme, în RFC 2821 s-a definit protocolul SMTP extins (**ESMTP**). Clienţii care doresc să-l utilizeze trebuie să trimită iniţial un mesaj *EHLO* în loc de *HELO.* Dacă acesta este rejectat, atunci serverul este unul standard de tip SMTP şi clientul va trebui să se comporte în modul obişnuit. Dacă *EHLO* este acceptat, înseamnă ca sunt permise noile comenzi şi noii parametri.

7.2.5 Livrarea finală

Până acum, am presupus că toţi utilizatorii lucrează pe maşini capabile să trimită şi să primească e-mail. După cum am văzut, e-mail-ul este livrat prin stabilirea unei conexiuni TCP între expeditor şi destinatar şi apoi prin trimiterea e-mail-ului prin ea. Acest model a funcţionat bine zeci de ani, atât timp cât toate calculatoarele din ARPANET (şi mai târziu din Internet) erau, de fapt, conectate la reţea şi gata să accepte conexiuni TCP.

Totuşi, odată cu apariţia celor care accesează Internet-ul folosind un modem cu care se conectează la ISP-ul lor, acest lucru nu mai ţine. Problema este următoarea: Ce se întâmplă când Elinor vrea să-i trimită Carolynei un e-mail şi Carolyn nu este conectată la reţea în acel moment? Elinor nu va putea să stabilească o conexiune TCP cu Carolyn şi astfel, nu va putea utiliza protocolul SMTP.

O soluţie este ca agentul de transfer de mesaje de pe o maşină ISP să accepte e-mail-ul pentru clienţii săi şi să-l stocheze în cutiile lor poştale pe o maşină a ISP-ului. Din moment ce acest agent poate fi conectat la reţea tot timpul, se poate trimite e-mail 24 de ore pe zi.

POP3

Din nefericire, această soluţie dă naştere altei probleme: cum îşi ia utilizatorul e-mail-ul de la agentul de transfer de mesaje al ISP-ului? Soluţia acestei probleme este crearea unui alt protocol care să permită agenţilor de transfer mesaje (aflaţi pe calculatoarele clienţilor) să contacteze agentul de transfer mesaje (de pe o maşină ISP) şi să facă posibilă copierea e-mail-ului de la ISP la utilizator. Un astfel de protocol este **POP3 (Post Office Protocol Version 3**- Protocol de poştă, versiunea 3**)**, definit în RFC 1939.

Situaţia anterioară (când atât expeditorul cât şi destinatarul aveau conexiune permanentă la Internet) este ilustrată în fig. 7-15(a). O situaţie în care expeditorul este efectiv conectat la reţea (online) dar destinatarul nu, este ilustrată în fig. 7-15(b).

POP3 începe când utilizatorul porneşte programul cititor de poştă (mail reader). Acesta sună la ISP (în cază că nu există deja o conexiune) şi stabileşte o conexiune

TCP cu agentul de transfer de mesaje, prin portul 110. Odată ce conexiunea a fost stabilită, protocolul POP3 trece succesiv prin următoarele trei stări:

1. Autorizare.

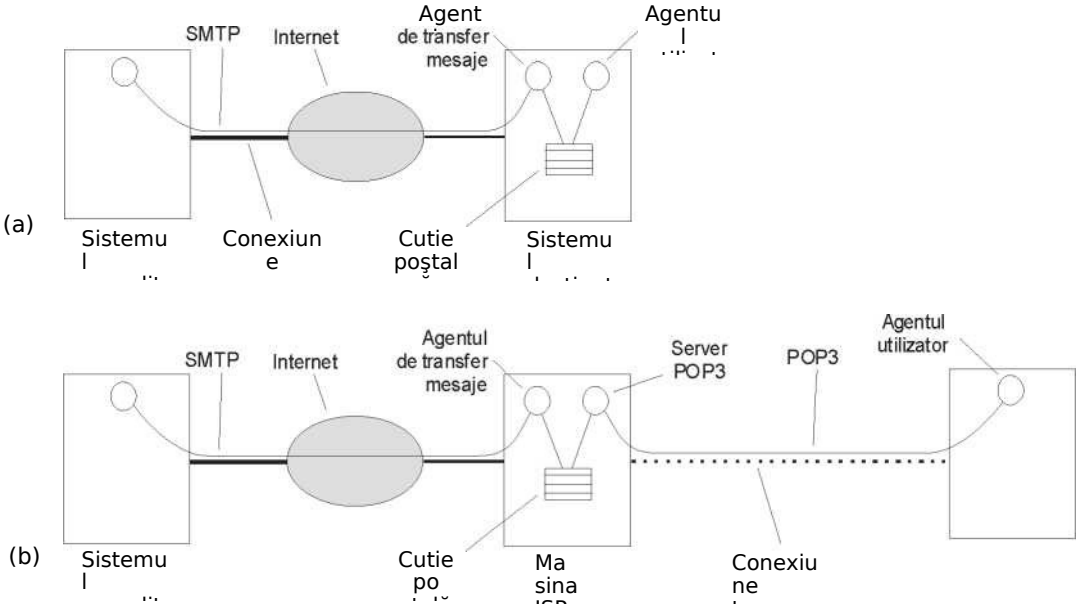


Fig. 7-15. (a) Trimiterea şi citirea poştei când destinatarul are o conexiune permanentă la Internet, iar agentul utilizator rulează pe aceeaşi maşină ca şi agentul de transfer de mesaje.

2. Tranzacţionare.
3. Actualizare.

Starea de autorizare se referă la admiterea utilizatorului în sistem (login). Starea de tranzacţionare tratează colectarea e-mail-urilor şi marcarea lor pentru ştergere din cutia poştală. Starea de actualizare se ocupă cu ştergerea efectivă a mesajelor.

Această comportare poate fi observată tastând ceva de genul:

```
telnet mail.isp.com 110
```

unde *mail.isp.com* reprezintă numele DNS al serverului de mail de la ISP. Telnet stabileşte o conexiune TCP prin portul 110, pe care ascultă serverul POP3. După acceptarea conexiunii TCP, serverul trimite un mesaj ASCII anunţându-şi prezenţa. De obicei, el începe cu +OK urmat de un comentariu. Un exemplu de scenariu este arătat în fig. 7-16 începând după ce conexiunea TCP a fost stabilită. Ca şi mai înainte, liniile marcate cu *C:* sunt ale clientului (utilizatorului) iar cele cu *S:* sunt ale serverului (agentul de transfer de mesaje de la ISP).

În timpul stării de autorizare, clientul trimite numele său de utilizator şi parola. După conectarea cu succes, clientul poate să trimită comanda *LIST,* care determină serverul să listeze conţinutul cutiei poştale. Lista este terminată cu un punct.

Apoi, clientul poate regăsi mesajele folosind comanda *RETR* şi le poate marca pentru ştergere cu *DELE.* Când toate mesajele au fost primite (şi eventual marcate pentru ştergere), clientul trimite comanda *QUIT* pentru terminarea stării de tranzacţionare şi intrarea în starea de actualizare. Când serverul a şters toate mesajele, el trimite un răspuns şi desfiinţează conexiunea TCP.

```
                    S: +OK Serverul POP3 este pregătit
C: USER carolyn
                    S: +OK
C: PASS vegetables
                    S: +OK autentificare cu succes
C: LIST
                     S: 1 2505 S: 2 14302 S: 3 8122 S: .
C: RETR 1
                    S: (trimite mesajul 1)
C: DELE 1
C: RETR 2
                    S: (trimite mesajul 2)
C: DELE 2
C: RETR 3
                    S: (trimite mesajul 3)
C: DELE 3
C: QUIT
                    S: +OK Serverul POP3 întrerupe legătura
```

Fig. 7-16. Folosirea protocolului POP3 pentru a aduce trei mesaje.

Deşi este adevărat că protocolul POP3 are abilitatea de a descărca un anumit mesaj sau un anumit grup de mesaje păstrându-le pe server, cele mai multe programe de e-mail descarcă tot şi golesc cutia poştală. Ca urmare, practic singura copie rămâne înregistrată pe discul utilizatorului. Dacă acesta se strică, toate e-mail-urile pot fi pierdute definitiv.

Să recapitulăm pe scurt cum lucrează e-mail-ul pentru clienţii unui ISP. Elinor creează un mesaj pentru Carolyn folosind un program de e-mail (adică, agentul utilizator) şi face clic pe o icoană pentru a-l trimite. Programul de e-mail trimite mesajul agentului de transfer de mesaje de pe calculatorul Elinorei. Agentul de transfer de mesaje vede că mail-ul este pentru *carolyn@xyz.com* şi foloseşte DNS pentru a căuta înregistrarea MX pentru *xyz.com* (unde *xyz.com* este ISP-ul Carolynei). Această cerere întoarce numele DNS al serverului de mail *xyz.com.* Agentul de transfer de mesaje caută acum adresa IP a acestei maşini folosind din nou DNS, de exemplu *gethostbyname.* Apoi, el stabileşte o conexiune TCP cu serverul SMTP pe portul 25 de pe această maşină. Folosind o secvenţă de comenzi SMTP, asemănătoare celei din fig. 7-14, el transferă mesajul în cutia poştală a Carolynei şi întrerupe conexiunea TCP.

După un timp, Carolyn îşi porneşte PC-ul său, se conectează la ISP şi porneşte programul de email. Programul de e-mail stabileşte o conexiune TCP cu serverul POP3 pe portul 110 al serverului de poştă al ISP-ului. Numele DNS sau adresa IP a acestei maşini este configurată în mod normal atunci când programul de e-mail este instalat sau când este făcut contractul cu ISP-ul. După ce conexiunea TCP a fost stabilită, programul de e-mail al Carolynei lansează protocolul POP3 pentru a aduce conţinutul cutiei sale poştale pe discul fix folosind comenzi ca cele din fig. 7-16. Odată ce a fost transferat tot e-mail-ul, conexiunea TCP este eliberată. De fapt, conexiunea cu ISP-ul poate fi desfiinţată acum, din moment ce tot e-mailul este pe discul fix al Carolynei. Desigur, pentru a trimite un răspuns, va fi nevoie din nou de

conexiunea cu ISP-ul, care, în general, nu este întreruptă imediat după aducerea poştei.

IMAP

Pentru un utilizator cu un singur cont de e-mail, la un singur ISP, care este tot timpul accesat de la un singur PC, POP3 este bun şi larg folosit datorită simplităţii şi robusteţii sale. Totuşi, există în industria calculatoarelor un adevăr bine înrădăcinat, acela că imediat ce un lucru funcţionează bine, cineva va începe să ceară mai multe facilităţi (şi să aibă mai multe probleme). Asta s-a întâmplat şi cu e-mail-ul. De exemplu, multă lume are un singur cont de e-mail la serviciu sau la şcoală şi vrea să-l acceseze de pe PC-ul de acasă, de pe calculatorul portabil în călătoriile de afaceri şi din Internet cafe-uri în vacanţe. Cu toate că POP3 permite asta, din moment ce în mod normal el descarcă toate mesajele la fiecare conectare, rezultatul constă în răspândirea e-mail-ului utilizatorului pe mai multe maşini, mai mult sau mai puţin la întâmplare, unele dintre ele nefiind ale utilizatorului.

Acest dezavantaj a dat naştere unei alternative a protocolului de livrare finală, **IMAP (Internet Message Access Protocol –** Protocol pentru accesul mesajelor în Internet**)**, care este definit în RFC 2060. Spre deosebire de POP3, care în mod normal presupune că utilizatorul îşi va goli căsuţa poştală la fiecare conectare şi va lucra deconectat de la reţea (off-line) după aceea, IMAP presupune că tot e- mail-ul va rămâne pe server oricât de mult, în mai multe căsuţe poştale. IMAP prevede mecanisme extinse pentru citirea mesajelor sau chiar a părţilor de mesaje, o facilitate folositoare când se utilizează un modem încet pentru citirea părţii textuale a unui mesaj cu mai multe părţi audio şi video de mari dimensiuni. Întrucât premisa de folosire este că mesajele nu vor fi transferate pe calculatorul utilizatorului în vederea stocării permanente, IMAP asigură mecanisme pentru crearea, distrugerea şi manipularea mai multor cutii poştale pe server. Astfel, un utilizator poate păstra o cutie poştală pentru fiecare corespondent şi poate muta aici mesajele din inbox după ce acestea au fost citite.

IMAP are multe facilităţi, ca de exemplu posibilitatea de a se referi la un mesaj nu prin numărul de sosire, ca în fig. 7-8, ci utilizând atribute (de exemplu, dă-mi primul mesaj de la Bobbie). Spre deosebire de POP3, IMAP poate de asemenea să accepte atât expedierea mesajelor spre destinaţie cât şi livrarea mesajelor venite.

Stilul general al protocolului IMAP este similar cu cel al POP3-ului, după cum se arată în fig. 716, cu excepţia faptului că există zeci de comenzi. Serverul IMAP ascultă pe portul 143. În fig. 7-17 este prezentată o comparaţie între POP3 şi IMAP. E bine de notat, totuşi, că nu toate ISP-urile oferă ambele protocoale şi că nu toate programele de e-mail le suportă pe amândouă. Aşadar, atunci când alegeţi un program de e-mail, este important să aflaţi ce protocoale suportă şi să vă asiguraţi că ISP-ul oferă cel puţin unul din ele.

Caracteristica	POP3	IMAP
Unde este definit protocolul	RFC1939	RFC 2060
Portul TCP folosit	110	143
Unde este stocat e-mail-ul	PC-ul utilizatorului	Server
Unde este citit e-mail-ul	Off-line	On-line
Timpul necesar conectării	Mic	Mare

Folosirea resurselor serverului	Minimă	Intensă
Mai multe cutii poştale	Nu	Da
Cine face copii de siguranţă la cutiile poştale	Utilizatorul	ISP-ul
Bun pentru utilizatorii mobili	Nu	Da
Controlul utilizatorului asupra scrisorilor	Mic	Mare
Descărcare parţială a mesajelor	Nu	Da
Volumul discului alocat (disk quota) este o	Nu	Ar putea fi în timp
Simplu de implementat	Da	Nu
Suport răspândit	Da	În creştere

Fig. 7-17. O comparaţie între POP3 şi IMAP.

Facilităţi de livrare

Indiferent dacă este folosit POP3 sau IMAP, multe sisteme oferă legături pentru procesarea adiţională a mesajelor e-mail sosite. Un instrument deosebit de valoros pentru mulţi utilizatori de email este reprezentat de capacitatea de a construi filtre. Acestea sunt reguli care se verifică la sosirea mesajelor sau la pornirea agentului utilizator. Fiecare regulă specifică o condiţie şi o acţiune. De exemplu, o regulă ar putea spune că orice mesaj venit de la şef trebuie pus în cutia poştală numărul 1, orice mesaj de la un anumit grup de prieteni se duce în cutia poştală numărul 2 şi orice alt mesaj conţinând anumite cuvinte în Subiect este aruncat fără comentarii.

Unii ISP oferă filtre care clasifică automat mesajele sosite ca fiind importante sau nerelevante (spam) şi memorează fiecare mesaj în cutia poştală corespunzătoare. Asemenea filtre funcţionează verificând mai întâi dacă sursa este un autor cunoscut de mesaje „spam”. Apoi examinează subiectul. Dacă sute de utilizatori au primit un mesaj cu acelaşi subiect, probabil că el este nerelevant. Există şi alte tehnici folosite pentru detecţia mesajelor lipsite de importanţă.

O altă caracteristică a livrării, pusă la dispoziţie adesea, este posibilitatea de a retrimite (temporar) poşta la o adresă diferită. Această adresă poate fi şi un calculator utilizat de un serviciu comercial de comunicaţii, care va contacta utilizatorul prin radio sau satelit, afişând *Subject:* linie pe pagerul său.

O altă trăsătură comună a livrării finale este abilitatea de a instala un **demon de vacanţă**. Acesta este un program care examinează fiecare mesaj sosit şi trimite o replică insipidă cum ar fi:

Salut. Sunt în vacanţă. Mă întorc pe 24 august. O zi bună.

Asemenea răspunsuri pot să specifice, de asemenea, cum să fie tratate problemele urgente, alte persoane care pot fi contactate pentru probleme specifice etc. Majoritatea demonilor de vacanţă păstrează urma celor cărora le-au trimis replici şi se abţin de la a trimite unei aceleiaşi persoane o a doua asemenea replică. Demonii buni verifică şi dacă mesajul sosit a fost trimis de la o listă de mail şi în acest caz, nu mai răspund deloc. (Cei care trimit mesaje în timpul verii la liste mari de email, probabil că nu doresc să primească sute de replici în care să le fie detaliate planurile de vacanţă ale fiecăruia.)

Autorul s-a lovit recent de o formă extremă de prelucrare a livrării când a trimis o scrisoare unei persoane care pretinde că primeşte 600 de mesaje pe zi. Identitatea sa nu va fi deconspirată aici, ca nu cumva jumătate dintre cititorii acestei cărţi să-i trimită şi ei scrisori. Să-l numim în continuare John.

John şi-a instalat un robot de e-mail care verifică fiecare mesaj sosit, ca să vadă

dacă este de la un nou corespondent. Dacă este aşa, trimite înapoi o replică standard în care explică faptul că nu mai poate să citească personal toate mesajele. În schimb a produs un document FAQ (Frequently Asked Questions) personal, unde răspunde la multe întrebări care i se pun de obicei. În mod normal, grupurile de ştiri şi nu persoanele au documente FAQ.

Documentul FAQ al lui John dă adresa acestuia, numărul de fax şi numerele de telefon şi spune cum poate fi contactată firma sa. Arată cum poate fi chemat ca vorbitor şi explică cum pot fi obţinute lucrările sale şi alte documente. Furnizează de asemenea referinţe la programele scrise de el, o conferinţă pe care o organizează, un standard al cărui editor este şi aşa mai departe. E posibil ca această abordare să fie necesară, dar poate că un FAQ personal reprezintă simbolul final al statutului.

Poşta electronică pe Web (Webmail)

Un subiect care merită menţionat este poşta electronică pe Web. Anumite situri de Web, cum ar fi Hotmail sau Yahoo oferă servicii de poştă electronică oricui doreşte. Ele funcţionează după cum urmează. Au agenţi normali de transfer de mesaje, care aşteaptă la portul 25 conexiuni noi de SMTP. Pentru a contacta, să spunem Hotmail, trebuie să obţineţi înregistrarea sa DNS *MX,* de exemplu tastând

```
host —a —v hotmail.com
```

pe un sistem UNIX. Să presupunem că serverul de poştă electronică se numeşte *mx10.hotmail.com;* atunci tastând

```
telnet mx10.hotmail.com 25
```

se poate stabili o conexiune TCP prin care se pot trimite comenzi SMTP în modul obişnuit. Deocamdată, nimic special, cu excepţia faptului că aceste servere mari sunt adeseori ocupate, ca atare se poate să dureze ceva mai mult până vă este acceptată o cerere de conexiune TCP.

Partea interesantă este cum se transmite poşta electronică. În principiu, atunci când utilizatorul se duce la pagina de Web a poştei electronice, îi este prezentat un formular în care i se cere un nume de cont şi o parolă. Când utilizatorul face clic pe **Sign În**, numele de cont şi parola sunt trimise serverului, care le validează. Dacă autentificarea s-a făcut cu succes, serverul găseşte cutia poştală a utilizatorului şi construieşte o listă similară cu cea din fig. 7-8, cu diferenţa că are formatul unei pagini de Web în HTML. Pagina Web este transmisă apoi programului de navigare pentru a fi afişată. Pe multe din elementele paginii se pot executa clic-uri, astfel că mesajele pot fi citite, şterse, ş.a.m.d.

7.3 WORLDWIDEWEB

Web-ul este un context arhitectural pentru accesul la documente, răspândite pe mii de maşini din Internet, între care există legături. În 10 ani a evoluat de la o aplicaţie pentru transmiterea de date utile pentru fizica energiilor înalte la o

aplicaţie despre care milioane de oameni cred că este Inter- netul. Popularitatea sa enormă se datorează faptului că are o interfaţă grafică plină de culoare, uşor de utilizat de către începători şi în acelaşi timp oferă o cantitate imensă de informaţie – de la animale mitologice la tribul Zulu, pe aproape orice subiect posibil.

Web-ul (cunoscut şi ca **WWW)** a apărut în 1989 la CERN, Centrul European de Cercetări Nucleare. CERN are câteva acceleratoare utilizate de echipe mari de cercetători din ţările europene pentru cercetări în fizica particulelor. Deseori aceste echipe au membri din peste zece ţări. Majoritatea experienţelor sunt foarte complicate şi necesită ani de pregătire şi construire de echipamente. Web-ul a apărut din necesitatea de a permite cercetătorilor răspândiţi în lume să colaboreze utilizând colecţii de rapoarte, planuri, desene, fotografii şi alte tipuri de documente aflate într-o continuă modificare.

Propunerea iniţială pentru crearea unei colecţii de documente având legături între ele (Web) a fost făcută de fizicianul Tim Berners-Lee, fizician la CERN, în martie 1989. Primul prototip (bazat pe text) era operaţional 18 luni mai târziu. În decembrie 1991, s-a făcut o demonstraţie publică la conferinţa Hypertex'91 în San Antonio, Texas.

Aceasta demonstraţie şi publicitatea aferentă au atras atenţia altor cercetători, fapt care l-a determinat pe Marc Andreessen de la University of Ilinois să înceapă să dezvolte primul program de navigare grafic, Mosaic. Acesta a fost lansat în februarie 1993. Mosaic a fost atât de popular încât un an mai târziu Marc Andreessen a plecat pentru a forma o nouă companie, Netscape Communications Corp., care se ocupa cu dezvoltarea de software pentru Web. Când Netscape a devenit o com
panie publică în 1995, investitorii, care probabil că au crezut că este vorba de un fenomen de tip Microsoft, au plătit 1,5 miliarde de dolari pentru acţiunile companiei. Acest record a fost cu atât mai neaşteptat cu cât compania avea un singur produs, opera în deficit şi anunţase probabilii investitori că nu se aşteaptă la beneficii în viitorul apropiat. În următorii trei ani, Netscape Navigator şi produsul Internet Explorer al companiei Microsoft au intrat într-un „război al programelor de navigare”, fiecare din produse încercând cu frenezie adăugarea de noi opţiuni (şi astfel a mai multor erori) decât celălalt. În 1998, America Online a cumpărat Netscape Communications Corp. pentru suma de
4.2 miliarde $, încheind astfel durata scurtă în care Netscape a fost o companie independentă.

În 1994, CERN şi M.I.T. au semnat o înţelegere pentru a forma **Consorţiul World Wide Web** (câteodată abreviat ca **W3C**), o organizaţie care are ca obiectiv dezvoltarea Web-ului, standardizarea protocoalelor, şi încurajarea interoperabilităţii între situri. Berners-Lee a devenit director. De atunci, sute de universităţi şi companii au intrat în consorţiu. M.I.T. coordonează partea americană a consorţiului în timp ce centrul de cercetări francez, INRIA, coordonează partea europeană. Deşi există foarte multe cărţi despre Web, cel mai bun loc pentru găsirea unor informaţii la zi despre el este (în mod natural) chiar Web-ul. Pagina consorţiului are adresa *www.w3.org* . Cititorii interesaţi vor găsi acolo legături la pagini care acoperă toate documentele şi activităţile consorţiului.

7.3.1 Aspecte arhitecturale

Din punctul de vedere al utilizatorului, Web-ul constă dintr-o colecţie imensă de documente sau **pagini de Web (Web pages)**, adesea numite prescurtat **pagini**, răspândite în toată lumea. Fiecare pagină poate să conţină legături (indicatori) la alte pagini, aflate oriunde în lume. Utilizatorii pot să aleagă o legătură prin execuţia unui clic care îi va aduce la pagina indicată de legătură. Acest proces se poate repeta la nesfârşit. Ideea ca o pagină să conţină legături către altele a fost inventată în 1945, cu mult înainte de a se fi inventat Internet-ul, de către Vannevar Bush, un profesor vizionar de la departamentul de inginerie electrică al M.I.T.

Paginile pot să fie văzute cu ajutorul unui **program de navigare** (**browser**). Internet Explorer şi Netscape Navigator sunt cele mai cunoscute programe de navigare. Programul de navigare aduce pagina cerută, interpretează textul şi comenzile de formatare conţinute în text şi afişează pagina, formatată corespunzător, pe ecran. Un exemplu este prezentat în fig. 7-18(a). Ca majoritatea paginilor de Web, începe cu un titlu, conţine informaţii şi se termină cu adresa de poştă electronică a celui care menţine pagina. Şirurile de caractere care reprezintă legături la alte pagini, se numesc **hiper-legături**, sunt afişate în mod diferit, fiind subliniate şi/sau colorate cu o culoare specială. Pentru a selecta o legătură, utilizatorul va plasa cursorul pe zona respectivă, ceea ce va determina schimbarea formei cursorului şi va executa un clic. Deşi există programe de navigare fără interfaţă grafică, ca de exemplu Lynx, ele nu sunt atât de utilizate ca programele de navigare grafice, astfel încât în continuare ne vom referi numai la ultimele. Au fost dezvoltate şi programe de navigare bazate pe voce.

Utilizatorii care sunt interesaţi să afle mai multe despre „Department of Animal Psichology” vor selecta numele respectiv (apare subliniat). Programul de navigare va aduce pagina la care este legat numele respectiv şi o va afişa, aşa cum se vede în fig. 7-18(b). Şirurile subliniate aici pot să fie selectate la rândul lor pentru a aduce alte pagini şi aşa mai departe. Noua pagină se poate afla pe aceeaşi maşină ca şi prima sau pe o maşină aflată undeva pe glob la polul opus. Utilizatorul nu va şti. Aducerea paginilor este realizată de către programul de navigare, fără nici un ajutor din partea utilizatorului. Dacă utilizatorul se va întoarce la prima pagină, legăturile care au fost deja utilizate vor fi afişate

WELLCOME TO THE UNIVERSITY OF EAST PODUNK'S WWW HOME PAGE

- Campus Information
  - Admissions information
  - Campus map
  - Directions to campus
  - The UEP student body

- Academic Departments
  - Department of Animal Psychology
  - Department of Alternative Studies
  - Department Microbiotic Cooking
  - Department Nontraditional Studies
  - Department of Traditional Studies

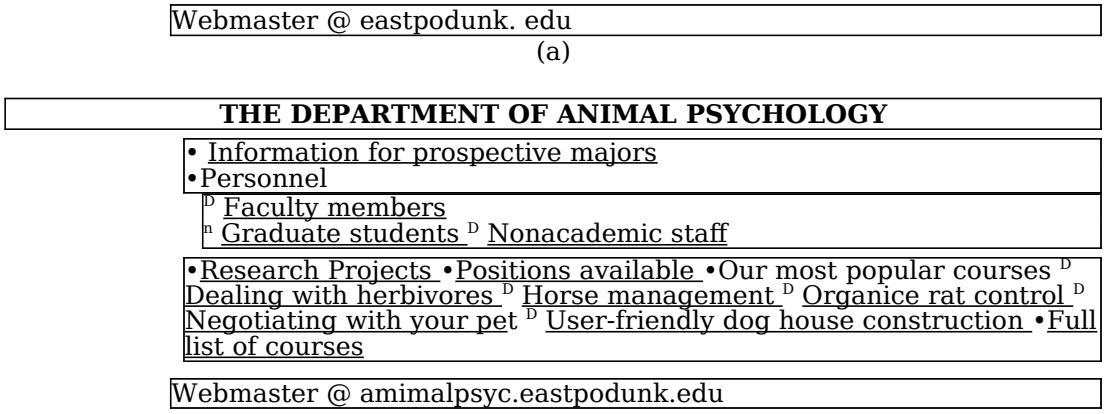

Webmaster @ eastpodunk. edu
(a)
THE DEPARTMENT OF ANIMAL PSYCHOLOGY
• Information for prospective majors
•Personnel
D Faculty members
n Graduate students D Nonacademic staff
•Research Projects •Positions available •Our most popular courses D
Dealing with herbivores D Horse management D Organice rat control D
Negotiating with your pet D User-friendly dog house construction •Full list of courses
Webmaster @ amimalpsyc.eastpodunk.edu
(b)

Fig. 7-18. (a) O pagină de Web. (b) pagina la care se ajunge dacă se selectează Department of Animal Psychology

altfel decât celelalte (subliniate cu linie punctată sau utilizând o altă culoare) pentru a fi deosebite de cele care nu au fost încă selectate. De notat că selecţia liniei *Campus Information* din prima pagină nu are nici un efect. Nu este subliniată, ceea ce înseamnă că este pur şi simplu un text care nu este legat de o altă pagină.

Modelul de bază al funcţionării Web-ului este arătat în fig. 7-19. Aici un program de navigare afişează o pagină de Web pe maşina clientului. Atunci când utilizatorul face clic pe linia de text ce indică spre o pagină de pe serverul *abcd.com,* programul de navigare urmează hiper-legătura trimiţând un mesaj serverului *abcd.com* în care se cere pagina respectivă. Atunci când pagina soseşte, ea este afişată. Dacă această pagină conţine o hiper-legătură către o pagină de pe serverul *xyz.com* pe care utilizatorul face clic, programul de navigare trimite o cerere maşinii respective pentru acea pagină, şi aşa mai departe la nesfârşit.

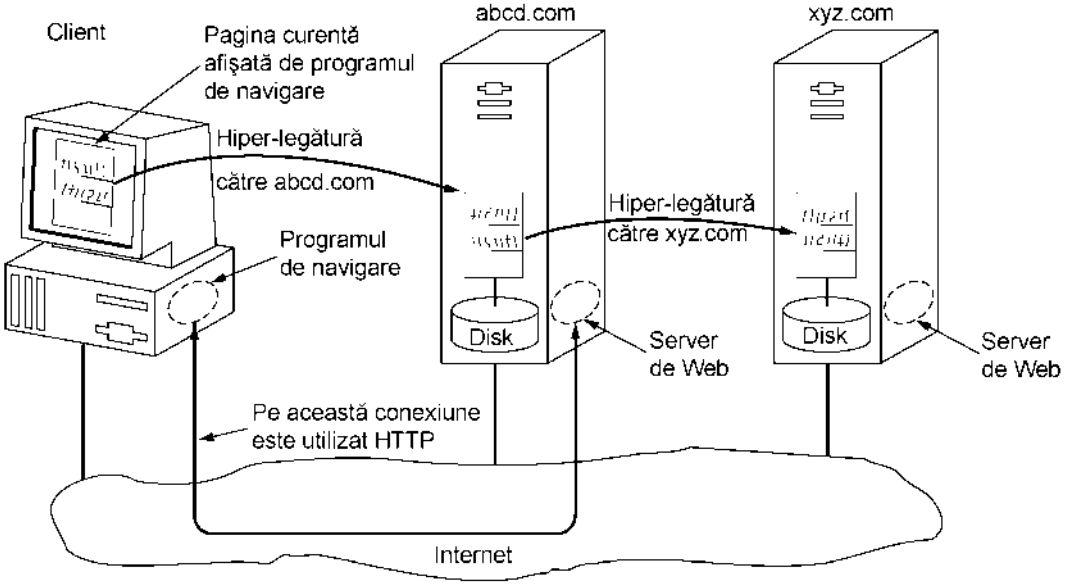


Fig. 7-19. Părţile componente ale modelului Web-ului

Aspecte privind clientul

Să examinăm acum în detaliu aspectele ce privesc clientul din fig. 7-19. În esenţă, un program de navigare este o aplicaţie capabilă să afişeze o pagină de Web şi să capteze clicurile mouse-ului pe elemente ale paginii afişate. Când un element este selectat, programul de navigare urmează hiper- legătura şi obţine pagina selectată. Ca atare, hiper-legătura conţinută în pagină necesită o modalitate de a adresa prin nume orice altă pagină de pe Web. Paginile sunt adresate prin nume folosind **URLuri (Uniform Resource Locators,** rom.: **Localizatoare Uniforme de Resurse)**. Un URL tipic este

http://www.abcd.com/products.html

Vom explica ce înseamnă URL mai târziu, în cadrul capitolului curent. Deocamdată, este suficient să ştim că un URL are trei părţi: numele protocolului *(http),* numele calculatorului pe care se găseşte pagina *(www.abcd.com)* şi numele fişierului care conţine pagina *(products.html).*

Când un utilizator execută un clic pe o hiper-legătură, programul de navigare urmează o serie de etape pentru a obţine pagina indicată de hiper-legătură. Să presupunem ca utilizatorul navighează pe Web şi găseşte o legătura despre telefonia pe Internet care indică spre pagina principală a ITU, *http:llwww.itu.orglhomelindex.html*. Să urmăm etapele parcurse când această legătură este selectată.

1. Programul de navigare determină URL (pe baza selecţiei).
2. Programul de navigare întreabă DNS care este adresa IP pentru *www.itu.org*.
3. DNS răspunde cu 156.106.192.32.
4. Programul de navigare realizează conexiunea TCP cu portul 80 al 156.106.192.32.
5. Trimite o cerere pentru fişierul *lhomelindex.html.*

6. Serverul *www.itu.org* transmite fişierul */homelindex.html.*
7. Conexiunea TCP este eliberată.
8. Programul de navigare afişează textul din */homelindex.html.*
9. Programul de navigare aduce şi afişează toate imaginile din acest fişier.

Multe programe de navigare informează despre etapa care se execută într-o fereastră de stare, în partea de jos a paginii. În acest mod, dacă performanţele sunt slabe, utilizatorul poate să ştie dacă este vorba de faptul că DNS nu răspunde, că serverul nu răspunde, sau pur şi simplu de congestia reţelei în timpul transmisiei paginii.

Pentru a afişa noua pagină (sau orice pagină), programul de navigare trebuie să înţeleagă forma în care este scrisă. Pentru a permite tuturor programelor de navigare să înţeleagă orice pagină de Web, paginile de Web sunt scrise într-un limbaj standardizat numit HTML, care descrie paginile de Web. Vom discuta acest limbaj mai târziu în acest capitol.

Deşi un program de navigare este în principiu un interpretor de HTML, majoritatea programelor de navigare au numeroase butoane şi opţiuni care ajută navigarea prin Web. Multe au un buton pentru revenirea la pagina anterioară, un buton pentru a merge la pagina următoare (acest buton este operaţional numai după ce utilizatorul s-a întors înapoi dintr-o pagină) şi un buton pentru selecţia paginii personale (home page). Majoritatea programelor de navigare au un buton sau un meniu pentru înregistrarea unei adrese de pagină (bookmark) şi un altul care permite afişarea unor adrese înregistrate, făcând astfel posibilă revenirea la o pagină cu ajutorul câtorva selecţii simple realizate cu mausul. Paginile pot să fie salvate pe disc sau tipărite. Sunt disponibile numeroase opţiuni pentru controlul ecranului şi configurarea programului de navigare conform dorinţelor utilizatorului.

În afară de text obişnuit (nesubliniat) şi hipertext (subliniat), paginile de Web pot să conţină iconiţe, desene, hărţi, fotografii. Fiecare dintre acestea poate să fie, în mod opţional, legată la altă pagin ă. Dacă se selectează unul dintre aceste elemente, programul de navigare va aduce pagina respectivă şi o va afişa, aşa cum se întâmplă în cazul selectării unui text. Pentru imaginile care sunt fotografii sau hărţi, alegerea paginii care se aduce poate să depindă de regiunea din imagine pe care se face selecţia.

Nu toate paginile conţin HTML. O pagină poate fi formată dintr-un document în format PDF, o iconiţă în format GIF, o fotografie în format JPEG, o melodie în format MP3, o înregistrare video în format MPEG sau oricare din cele alte câteva sute de tipuri de fişiere. Deoarece paginile în forma standard HTML pot avea legături către oricare din acestea, programul de navigare are o problemă atunci când întâlneşte o pagină pe care nu o poate interpreta.

În loc sa facă programele de navigare din ce în ce mai mari, înglobând interpretoare pentru o colecţie de tipuri de fişiere în creştere rapidă, majoritatea programelor de navigare au ales o soluţie mai generală. Atunci când un server întoarce o pagină, el întoarce de asemenea informaţii adiţionale despre acea pagină. Această informaţie include tipul MIME al paginii (fig. 7-12). Paginile de tipul *text/html* sunt afişate direct, ca şi paginile de alte câteva tipuri interpretate implicit. Dacă tipul MIME nu este unul dintre acestea, programul de navigare îşi consultă

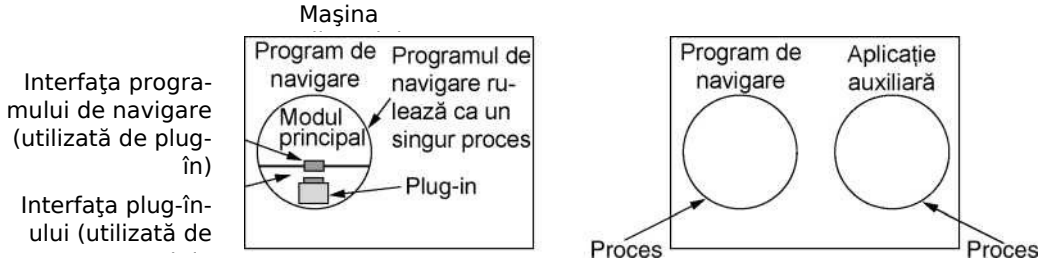


tabela de tipuri MIME pentru a afla cum să afişeze pagina. Această tabelă asociază un tip MIME cu o aplicaţie de vizualizare.

Fig. 7-20

Există două posibilităţi: plug-în-uri şi programe auxiliare *(helper applications).* Un **plug-în** este un modul pe care programul de navigare îl obţine dintr-un director special de pe disc şi îl instalează ca o extensie al însuşi programului de navigare, aşa cum se arată în fig. 7-20(a). Deoarece plug-în-urile se execută în interiorul programului de navigare, acestea au acces la pagina curentă şi pot să modifice modul în care aceasta este afişată. După ce plug-în-ul a terminat ceea ce avea de făcut (de obicei după ce utilizatorul s-a deplasat la altă pagină de Web), acesta este scos din memoria programului de navigare.

Fiecare program de navigare are o colecţie de proceduri pe care toate plug-în-urile trebuie să le implementeze pentru ca programul de navigare să poată executa plug-în-ul. De exemplu, există de obicei o procedură pe care modulul principal al programului de navigare o apelează pentru a oferi plug-în-ului date ce trebuie afişate. Această colecţie de proceduri constituie interfaţa unui plug-în şi este particulară fiecărui program de navigare.

În plus, programul de navigare pune la dispoziţia plug-în-ului propria sa colecţie de proceduri. Procedurile tipice din interfaţa programului de navigare se referă la alocarea şi eliberarea memoriei, afişarea de mesaje în fereastra de stare a programului de navigare sau obţinerea parametrilor acestuia.

Înainte ca un plug-în să poată fi folosit, acesta trebuie instalat. Procedura uzuală de instalare este ca utilizatorul să navigheze la situl de Web al plug-în-ului şi să copieze un fişier de instalare. Pe sistemele Windows, acesta este de obicei o arhivă zip cu decompresie automată cu extensia *.exe.* Când pe acest fişier se execută un dublu clic, se execută un program de mici dimensiuni ataşat părţii de început a fişierului. Acest program decomprimă plug-în-ul şi îl copiază în directorul de plug-în-uri al programului de navigare. Apoi execută apelurile necesare pentru înregistrarea tipului MIME corespunzător şi asocierea acestuia cu plug-în-ul. Pe sistemele UNIX, fişierul de instalare este în mod frecvent un fişier de comenzi care se ocupă de copiere şi înregistrare.

Cea de-a doua modalitate de extindere a unui program de navigare este utilizarea **aplicaţiilor auxiliare (helper applications)**. Acestea sunt programe complete ce se execută ca procese separate. Acest fapt este ilustrat în fig. 7-20(b). Deoarece acestea sunt programe separate, nu oferă nici o interfaţă programului de navigare şi nu utilizează serviciile acestuia. De obicei însă acceptă doar numele unui fişier temporar unde a fost stocat conţinutul paginii, deschide acest fişier şi îi afişează conţinutul. De obicei, aplicaţiile auxiliare sunt programe de dimensiuni mari care există independent de programul de navigare, cum ar fi Adobe Acrobat Reader pentru afişarea fişierelor PDF, sau Microsoft Word. Unele programe (cum ar fi Acrobat) dispun de un plug-în care execută aplicaţia auxiliară.

Multe aplicaţii auxiliare folosesc tipul MIME *application.* A fost definit un număr considerabil de subtipuri, de exemplu *application/pdf* pentru fişiere PDF şi *application/msword* pentru fişiere Word. În acest mod, un URL poate să indice direct către un fişier PDF sau Word şi atunci când utilizatorul execută un clic asupra

sa aplicaţiile Acrobat sau Word sunt pornite automat şi li se transmite numele fişierului temporar ce conţine datele ce trebuie afişate. Ca atare, programele de navigare pot fi configurate să trateze un număr teoretic nelimitat de tipuri de documente, fără schimbări aduse programului de navigare. Serverele de Web moderne sunt adesea configurate cu sute de combinaţii de tipuri/subtipuri şi combinaţii noi sunt adăugate de fiecare dată când este instalat un program nou.

Aplicaţiile auxiliare nu sunt restricţionate la utilizarea tipului MIME *application*. Adobe Photoshop foloseşte *image/x-photoshop* şi RealOne Player poate trata de exemplu *audio/mp3.*

Pe sistemele Windows, atunci când un program este instalat pe un calculator, el înregistrează tipurile MIME pe care doreşte să le trateze. Acest mecanism conduce la conflicte atunci când mai multe aplicaţii sunt disponibile pentru vizualizarea unui subtip, cum ar fi *video/mpg.* Ceea ce se întâmplă este că ultimul program ce se înregistrează supraînscrie asociaţiile existente (tip MIME, aplicaţie auxiliară), captând tipul pentru sine. Ca o consecinţă, instalarea unui nou program poate schimba modul în care un program de navigare tratează tipurile existente.

Pe sistemele UNIX, acest proces de înregistrare nu se face în general automat. Utilizatorul trebuie să schimbe anumite fişiere de configurare. Această abordare conduce la un volum mai mare de muncă dar la surprize mai puţine.

Programele de navigare pot de asemenea deschide fişiere locale în loc de a le obţine de pe servere de Web de la distanţă. Deoarece fişierele locale nu au tipuri MIME, programul de navigare necesită o metodă pentru a determina ce plug-în sau aplicaţie auxiliară trebuie folosit pentru alte tipuri decât cele tratate implicit ca *textlhtml* sau *image/jpeg.* Pentru tratarea fişierelor locale, aplicaţiile auxiliare pot fi asociate şi cu o extensie de fişier, ca şi cu un tip MIME. Considerând configuraţia standard, deschiderea fişierului *foo.pdf* îl va încărca în Acrobat şi deschiderea fişierului *bar.doc* îl va încărca în Word. Anumite programe de navigare folosesc tipul MIME, extensia fişierului şi chiar informaţii din interiorul fişierului pentru a ghici tipul MIME. În special Internet Explorer se bazează în primul rând pe extensia fişierului decât pe tipul MIME atunci când acest lucru este posibil.

Şi aici pot apare conflicte deoarece multe programe sunt dispuse, de fapt dornice să trateze *mpg*, de exemplu. În timpul instalării, programele create pentru profesionişti afişează frecvent căsuţe de selecţie pentru tipurile MIME şi extensiile pe care sunt pregătite să le trateze pentru a permite utilizatorului selecţia celor dorite pentru a preveni astfel supraînscrierea accidentală a asociaţiilor existente. Programele ce au ca ţintă marea masă a consumatorilor presupun că utilizatorul nu ştie ce este un tip MIME şi pur şi simplu acaparează tot ce pot fără să ţină seama de ceea ce au făcut programele instalate anterior.

Capacitatea de a extinde programul de navigare cu un număr mare de tipuri noi este utilă, dar poate duce şi la probleme. Atunci când Internet Explorer obţine un fişier cu extensia *exe* îşi da seama că acest fişier este un program executabil şi ca atare nu are o aplicaţie adiţională asociată. Acţiunea evidentă este execuţia fişierului. Însa această acţiune poate fi o problemă de securitate enormă. Tot ceea ce trebuie să facă un site de Web rău intenţionat este să ofere o pagină de Web, de exemplu cu fotografii de staruri de cinema sau sportive, toate imaginile indicând

către un virus. Un singur clic pe o imagine determină ca un program executabil necunoscut şi posibil ostil să fie copiat şi executat pe maşina utilizatorului. Pentru a preveni astfel de vizitatori nedoriţi, Internet Explorer poate fi configurat să fie selectiv în a executa programe necunoscute în mod automat, dar nu toţi utilizatorii înţeleg cum să folosească această configurare.

Pe sistemele UNIX pot exista probleme similare cu fişierele de comenzi pentru consola sistemului, dar aceasta necesită ca utilizatorul să instaleze în mod conştient consola sistemului ca aplicaţie auxiliară. Din fericire, acest proces este suficient de complicat pentru ca nimeni să nu poată să îl efectueze accidental (şi puţine persoane pot să îl efectueze chiar intenţionat).

Aspecte privind serverul

Cam atât despre aspectele privind clientul. Să ne referim acum la aspectele privind serverul. Aşa cum am văzut mai sus, atunci când utilizatorul tastează un URL sau execută un clic asupra unei linii de hipertext, programul de navigare analizează URL-ul şi interpretează partea între *http://* şi următorul caracter */* ca un nume DNS ce trebuie căutat. Înarmat cu adresa IP a serverului, programul de navigare stabileşte o conexiune TCP la portul 80 de pe acel server. Apoi se transmite o comandă ce conţine restul URL-ului, care este de fapt numele fişierului de pe acel server. Serverul întoarce apoi fişierul pentru a fi afişat de către programul de navigare.

Într-o primă aproximare, un server de Web este similar cu serverul din fig. 6-6. Acel server, ca şi un server de Web real, primeşte numele fişierului ce trebuie căutat şi transmis programului de navigare. În ambele cazuri, etapele pe care le parcurge serverul în bucla sa principală sunt:

1. Acceptă o conexiune TCP de la un client (program de navigare).
2. Obţine numele fişierului cerut.
3. Obţine fişierul (de pe disc).
4. Întoarce fişierul clientului.
5. Eliberează conexiunea TCP.

Serverele de Web moderne au mai multe caracteristici, dar în esenţă acestea sunt funcţiile unui server de Web. O problemă cu această arhitectură este că fiecare cerere necesită acces la disc pentru obţinerea fişierului. Rezultatul este că serverul de Web nu poate servi mai multe cereri pe secundă decât numărul de accese la disc ce se pot executa pe secundă. Un disc SCSI are un timp de acces mediu de circa 5 ms, ceea ce limitează serverul la cel mult 200 de cereri/sec, chiar mai puţin dacă trebuie citite des fişiere mari. Pentru un sit de Web de importanţă mare, acest număr este prea mic.

O îmbunătăţire evidentă (utilizată de toate serverele de Web) este folosirea unui sistem de memorie ascunsă, temporară pentru cele mai recente *n* fişiere utilizate. Înainte de obţinerea unui fişier de pe disc, serverul verifică memoria ascunsă (cache). Dacă fişierul există acolo, el poate fi servit direct din memorie, eliminând astfel accesul la disc. Deşi pentru o memorie ascunsă eficientă sunt necesare o cantitate mare de memorie principală şi timp de procesare suplimentară pentru a analiza memoria ascunsă şi pentru a-i administra conţinutul, economia de timp este

aproape întotdeauna superioară timpului suplimentar de procesare şi costului memoriei.

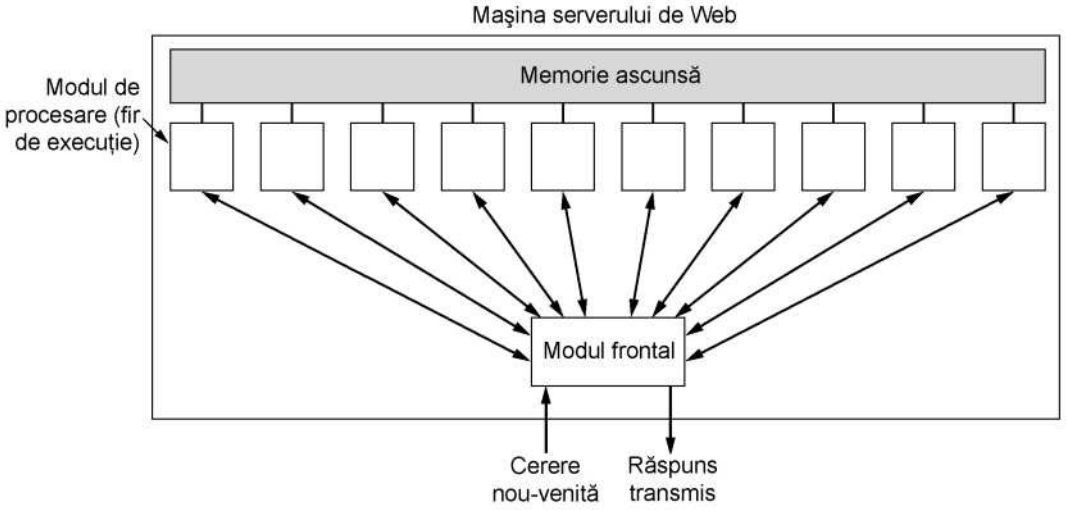


Fig. 7-21. Un server de Web cu mai multe fire de execuţie cu un modul frontal şi module de procesare

Următorul pas pentru construcţia unui server mai rapid este de a face serverul să admită mai multe fire de execuţie *(multithreaded).* Într-una din arhitecturi, serverul este format dintr-un modul frontal *(front-end module),* care acceptă conexiunile nou venite, şi k module de procesare, aşa cum arata fig. 7-21. Cele $k + 1$ fire de execuţie aparţin toate aceluiaşi proces, astfel că modulele de procesare au toate acces la memoria ascunsă din interiorul spaţiului de adrese al procesului. La sosirea unei cereri, modulul frontal o acceptă şi construieşte o scurtă înregistrare ce descrie cererea. Aceasta este transmisă apoi unuia dintre modulele de procesare. În altă arhitectură posibilă, modulul frontal este eliminat şi fiecare modul de procesare încearcă să îşi obţină propriile cereri, dar în acest caz este necesar un protocol de sincronizare pentru prevenirea conflictelor.

Modulul de procesare verifică mai întâi memoria ascunsă pentru a determina dacă fişierul necesar se află acolo. Dacă da, modifică înregistrarea pentru a include şi un indicator către fişierul din înregistrare. Dacă fişierul nu se află acolo, modulul de procesare începe operaţiile cu discul pentru a citi fişierul în memoria ascunsă (renunţând eventual la alte fişiere pentru a face loc acestuia). Când fişierul este citit de pe disc, el este pus în cache şi de asemenea transmis clientului.

Avantajul acestei scheme este că în timp ce unul sau mai multe module de procesare sunt blocate aşteptând terminarea operaţiilor cu discul (şi deci nu consumă din timpul procesorului), alte module pot fi active lucrând la satisfacerea altor cereri. Desigur, pentru a obţine o îmbunătăţire reală asupra modelului cu un singur fir de execuţie este necesară existenţa mai multor unităţi de disc, astfel încât mai multe discuri să poată fi ocupate în acelaşi timp. Cu ajutorul a k module de procesare şi k unităţi de disc, eficienţa poate creşte până la de k ori faţă de modelul serverului cu un singur fir de execuţie şi o singură unitate de disc.

Teoretic, un server cu un singur fir de execuţie şi *k* unităţi de disc poate de asemenea câştiga un factor *k* în ceea ce priveşte eficienţa, dar implementarea şi administrarea sunt mult mai complicate deoarece apelurile de sistem READ normale, blocante nu pot fi folosite pentru accesul la disc. În cazul unui server cu mai multe fire de execuţie, acestea pot fi folosite deoarece o operaţie READ blochează doar firul de execuţie care a executat operaţia şi nu întregul proces.

Serverele de Web moderne efectuează mai multe operaţii decât acceptarea numelor de fişiere şi transmiterea conţinutului acestora. De fapt, procesarea fiecărei cereri poate deveni destul de complicată. Din acest motiv, într-un număr mare de servere fiecare modul de procesare efectuează o serie de etape. Modulul frontal transmite fiecare cerere sosită către primul modul de procesare disponibil, care apoi execută cererea, utilizând o submulţime a următorilor paşi, în funcţie de ce paşi sunt necesari pentru respectiva cerere.

1. Rezolvarea numelui paginii de Web cerute.
2. Autentificarea clientului.
3. Verificarea drepturilor de acces ale clientului.
4. Verificarea drepturilor de acces asupra paginii de Web.
5. Verificarea memoriei ascunse.
6. Obţinerea paginii cerute, de pe disc.
7. Determinarea tipului MIME ce va fi inclus în răspuns.
8. Rezolvarea altor probleme minore.
9. Transmiterea răspunsului către client.
10. Adăugarea unei înregistrări în jurnalul serverului.

Pasul 1 este necesar deoarece cererea sosită poate să nu conţină numele propriu-zis al fişierului, ca şir de caractere. De exemplu, putem considera URL-ul *http:llwww.cs.vu.nl,* care are un nume de fişier vid. Acesta trebuie extins la un nume de fişier implicit. De asemenea, programele de navigare moderne pot specifica limba implicită a utilizatorului (de ex.: italiană sau engleză), ceea ce deschide posibilitatea ca serverul să selecteze o pagină de Web în acea limbă, dacă aceasta este disponibilă. În general, extinderea numelor nu este un proces atât de banal cum ar putea părea la prima vedere, datorită unei varietăţi de convenţii existente privind numirea fişierelor.

Pasul 2 constă în verificarea identităţii clientului. Acest pas este necesar pentru paginile care nu sunt disponibile publicului larg. Vom discuta o modalitate de a realiza acest lucru mai târziu, în acest capitol.

Pasul 3 verifică dacă există restricţii referitoare la satisfacerea cererii, având în vedere identitatea şi localizarea clientului. Pasul 4 verifică dacă există restricţii de acces asociate cu pagina însăşi. Dacă un anumit fişier (de ex.: *.htaccess)* este prezent în directorul unde se află şi pagina dorită, accesul la acel fişier poate fi restrâns la anumite domenii, de exemplu numai la utilizatorii din interiorul companiei.

Paşii 5 şi 6 presupun obţinerea paginii. Pasul 6 necesită capacitatea de tratare simultană a mai multor citiri de pe disc.

Pasul 7 se referă la determinarea tipului MIME din extensia fişierului, primele câteva cuvinte din fişier, un fişier de configurare sau alte surse posibile. Pasul 8 este destinat unei diversităţi de operaţii, cum ar fi construcţia unui profil al utilizatorului sau adunarea unor statistici.

Pasul 9 este cel în care rezultatul este transmis clientului şi pasul 10 adaugă o înregistrare în jurnalul sistemului, în scopuri administrative. Asemenea fişiere de jurnalizare pot fi analizate ulterior pentru obţinerea de informaţii importante despre comportamentul utilizatorului, spre exemplu ordinea în care vizitatorii accesează paginile.

Dacă sosesc prea multe cereri în fiecare secundă, procesorul nu va fi capabil sa suporte încărcarea, oricâte unităţi de disc ar fi utilizate în paralel. Soluţia este adăugarea mai multor noduri (calculatoare), posibil cu unităţi de disc replicate pentru a evita ca discurile să devină următorul punct de gâtuire. Acest fapt conduce la modelul **fermei de servere (server farm)** din fig. 7-22. Un modul frontal acceptă în continuare cererile dar le împarte mai multor procesoare, nu mai multor fire de execuţie, pentru a reduce încărcarea pe fiecare calculator. Maşinile individuale pot fi cu mai multe fire de execuţie şi în bandă de asamblare ca mai sus.

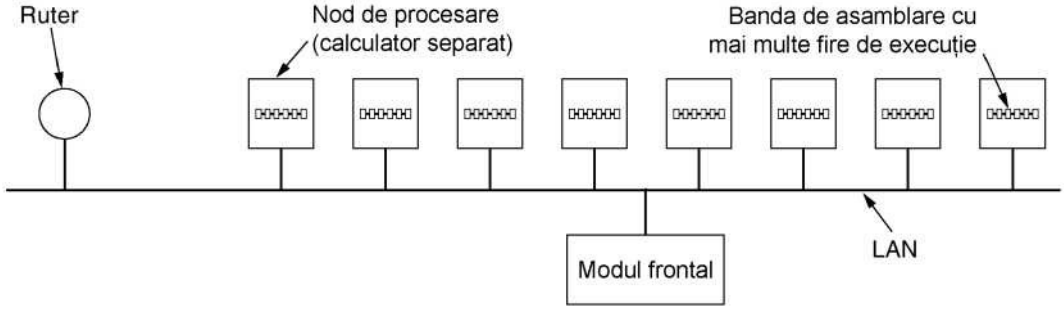


Fig. 7-22. O fermă de servere

O problemă cu fermele de servere este că nu mai există o memorie ascunsă partajată deoarece fiecare nod are propria sa memorie – decât dacă se foloseşte un sistem multiprocesor cu memorie partajată de cost mic. O modalitate de a contracara această pierdere de performanţă este un modul frontal care reţine unde direcţionează fiecare cerere şi trimite cererile ulterioare pentru aceeaşi pagină aceluiaşi nod. Această abordare specializează fiecare nod în tratarea anumitor pagini astfel încât spaţiul destinat pentru cache nu se pierde reţinând fiecare fişier în fiecare cache.

O altă problemă cu fermele de servere este aceea că fiecare conexiune TCP a clientului se termină la modulul de intrare, astfel că răspunsul trebuie transmis prin acest modul. Această situaţie este evidenţiată în fig. 7-23(a), unde cererea sosită (1) şi răspunsul transmis (4) trec ambele prin modulul frontal. Câteodată se poate folosi o soluţie ingenioasă numită **parsare TCP (eng.: TCP handoii)** pentru a evita această problemă. Cu acest truc, capătul comunicaţiei TCP este „pasat” nodului de procesare astfel că acesta poate replica direct clientului, lucru evidenţiat ca (3) în fig. 7-23(b). Această pasare este făcută într-un mod transparent pentru client.

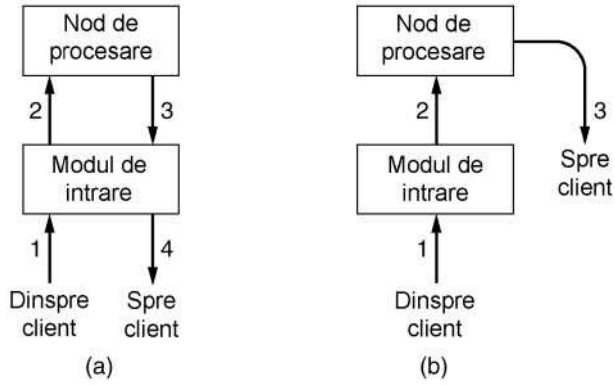


Fig. 7-23. (a) Secvenţă normală de mesaje cerere – răspuns. (b) Secvenţă în care se foloseşte pasarea TCP.

URL- Uniform Resource Locators

Am spus de mai multe ori că o pagină de Web poate să conţină referinţe la alte pagini. Să explicăm cum sunt implementate aceste referinţe. Încă de la crearea Web-ului, a fost clar că pentru a avea o pagină care să indice spre altă pagină este necesar un mecanism care să permită numirea şi regăsirea paginilor. În particular, sunt trei întreb ări la care trebuie să se răspundă înainte de a se putea afişa o pagină:

1. Cum se numeşte pagina?
2. Cum este localizată pagina?
3. Cum se face accesul la pagină?

Dacă fiecare pagină ar avea un nume unic, atunci nu ar exista nici o ambiguitate în identificarea paginilor. Totuşi, problema nu este încă rezolvată. Să considerăm de exemplu o paralelă între oameni şi pagini. În SUA aproape fiecare persoană are un număr de asigurare socială, care este un identificator unic, astfel încât nu există două persoane cu acelaşi număr. Totuşi, cunoscând numai numărul respectiv, nu există nici o posibilitate de a găsi adresa persoanei respective şi sigur nu se poate afla dacă persoanei respective trebuie să i se scrie în Engleză, Spaniolă sau Chineză. Web-ul are practic acelaşi fel de probleme.

Soluţia aleasă identifică paginile într-un mod care rezolvă toate trei problemele în acelaşi timp. Fiecare pagină are un **URL** (**Uniform Resource Locator –** adresa uniformă pentru localizarea resurselor) care funcţionează ca nume al paginii general valabil. Un URL are trei componente: protocolul (cunoscut şi sub numele de **schemă**), numele DNS al maşinii pe care este memorat fişierul şi un nume local, care indică în mod unic pagina (de obicei numele fişierului care conţine pagina). De exemplu, situl de Web al departamentului din care face parte autorul conţine un număr de înregistrări video despre universitate şi despre oraşul Amsterdam. URL-ul paginii cu înregistrările video este:

http://www.cs.vu.nl/video/index-en.html

Acest URL este format din trei componente: protocolul (*http*), numele DNS al serverului *(www. cs.vu.nl)* şi numele fişierului *(video/index-en.html),* cu semnele de

punctuaţie corespunzătoare. Numele fişierului este o cale relativă la directorul de Web implicit de la *cs.vu.nl.*

Se utilizează notaţii care reprezintă prescurtări standard. În cazul multor situri, un nume de fişier nul înseamnă implicit pagina principală a organizaţiei. În mod obişnuit, atunci când numele fişierului denotă un director, aceasta implică un fişier numit *index.html.* În sfârşit, *~user/* poate să fie pus în corespondenţă cu directorul WWW al utilizatorului *user*, şi apoi cu fişierul *index.html* în acest director. De exemplu, pagina autorului poate să fie referită ca:

http://www.cs.vu.nl/~ast/

chiar dacă de fapt numele propriu-zis al fişierului este *indexhtml,* implicit în acest director.

Acum ar trebui să fie clar cum funcţionează hipertextul. Pentru a face o porţiune de text selecta- bilă, cel care scrie pagina trebuie să furnizeze două elemente: textul prin care se face selecţia şi URL-ul paginii care trebuie adusă, dacă textul este selectat. Vom explica sintaxa comenzii mai târziu în acest capitol.

Când se face selecţia, programul de navigare caută numele serverului utilizând DNS-ul. Pe baza adresei IP a serverului, programul de navigare stabileşte o conexiune TCP spre server. Utilizând această conexiune, se transmite numele fişierului utilizând protocolul specificat. Bingo. Acum soseşte pagina.

Această schemă URL este deschisă în sensul că este simplu să se utilizeze alte protocoale pentru a se obţine diferite tipuri de resurse. De fapt au fost definite URL-uri pentru protocoalele obişnuite, şi multe programe de navigare înţeleg aceste protocoale. Forme simplificate ale celor mai obişnuite sunt prezentate în fig. 7-24.

Nume	Utilizat pentru	Exemple
http	Hipertext (HTML)	http://www.cs.vu.nl/~ast
ftp	FTP	ftp://ftp.cs.vu.nl/pub/minix/README
File	Fişier local	file:///usr/suzanne/prog.c
news	Grup de ştiri	news:AA0134223112@cs.utah.edu
news	Articol de ştiri	news:AA0134223112@cs.utah.edu
gopher	Gopher	gopher://gopher.tc.umn.edu/11/
mailto	Trimitere de poşta	mailto:JohnUser@acm.org
telnet	Conectare la distanţă	telnet://www.w3.org:80

Fig. 7-24. Câteva URL-uri obişnuite.

Să parcurgem lista rapid. Protocolul *http* este protocolul nativ pentru Web, el este utilizat de către serverele de Web. **HTTP** este o prescurtare pentru **HyperText Transfer Protocol**. Vom examina mai detaliat acest protocol mai târziu în acest capitol.

Protocolul *ftp* este utilizat pentru accesul la fişiere prin FTP (File Transfer Protocol – protocol pentru transferul de fişiere), protocolul Internet de transfer de fişiere. FTP este utilizat de peste douăzeci de ani şi este foarte răspândit. Numeroase servere de FTP din toată lumea permit ca de oriunde din Internet să se facă o conectare şi să se aducă orice fişier plasat pe un server FTP. Web-ul nu aduce schimbări aici, face doar ca obţinerea fişierelor să se facă mai uşor, pentru că FTP are o interfaţă mai puţin prietenoasă (dar este mai puternic decât HTTP, deoarece

permite de exemplu ca un utilizator de pe maşina A să transfere un fişier de pe maşina B pe maşina C).

Este posibil să se facă acces la un fişier local ca la o pagină de Web, fie utilizând protocolul *file* (fişier*)*, fie pur şi simplu utilizând numele fişierului. Această abordare este similară utilizării protocolului FTP, dar nu implică existenţa unui server. Desigur funcţionează numai pentru fişiere locale, nu şi pentru cele aflate la distanţă.

Cu mult înainte de apariţia Internet-ului exista sistemul de ştiri USENET. Acesta este format din aproximativ 30000 de grupuri de ştiri în care milioane de persoane discută despre o mare varietate de subiecte, adăugând şi citind articole legate de subiectul grupului de ştiri. Protocolul *news* permite citirea un articol din ştiri ca şi cum ar fi o pagină de Web. Aceasta înseamnă că un program de navigare este în acelaşi timp şi un cititor de ştiri. De fapt, multe programe de navigare au butoane sau elemente de meniu care permit citirea ştirilor USENET mai uşor decât dacă se utilizează cititoare standard de ştiri.

Protocolul *news* admite două formate. Primul format specifică un grup de ştiri şi poate să fie utilizat pentru a obţine o listă de articole de la un server de ştiri preconfigurat. Al doilea format cere identificatorul unui articol, de exemplu *AA0134223112@cs.utah.edu*. Programul de navigare aduce articolul de la serverul corespunzător utilizând protocolul **NNTP (Network News Transfer Protocol – Protocol de transfer al ştirilor prin reţea)**. Nu vom studia NNTP în această carte, dar este în mare bazat pe SMTP şi are un stil similar.

Protocolul *gopher* era utilizat de sistemul Gopher, care a fost proiectat la universitatea Minnesota. Numele este cel al echipei atletice a universităţii, the Golden Gopher (de asemenea acest nume este utilizat în argou pentru „go for” adică o comandă de aducere). Gopher-ul a precedat Web-ul cu câţiva ani. Era o metodă de regăsire a informaţiei, similară conceptual cu cea utilizată de Web, dar acceptând numai text şi imagini. Este considerat depăşit şi nu se mai foloseşte în prezent.

Ultimele două protocoale nu sunt de fapt protocoale pentru aducerea unor pagini de Web, dar sunt utile. Protocolul *mailto* permite transmiterea de poştă dintr-un program de navigare. Pentru a face această operaţie, se selectează butonul OPEN şi se specifică un URL constând din *mailto:* urmat de adresa destinatarului. Majoritatea programelor de navigare vor răspunde prin pornirea unei aplicaţii de poştă electronică cu adresa şi câteva alte câmpuri din antet deja completate.

Protocolul *telnet* este utilizat pentru stabilirea unei conexiuni cu o maşină aflată la distanţă. Se utilizează în acelaşi fel ca şi programul telnet, ceea ce nu constituie o surpriză, deoarece majoritatea programelor de navigare utilizează programul telnet ca aplicaţie auxiliară.

Pe scurt URL-urile au fost proiectate nu numai pentru a permite utilizatorilor să navigheze prin Web, dar şi pentru a utiliza FTP, news, Gopher, e-mail şi telnet, ceea ce face inutile interfeţele specializate pentru aceste protocoale integrând astfel într-un singur program, navigatorul în Web, aproape toate tipurile de acces în Internet. Dacă metoda nu ar fi fost proiectată de un fizician, ar fi putut să pară produsul departamentului de publicitate al unei companii de software.

În ciuda tuturor acestor proprietăţi, creşterea Web-ului scoate în evidenţă şi o slăbiciune a metodei utilizării URL-urilor. Pentru o pagină care este foarte des

referită, ar fi de preferat să existe mai multe copii pe servere diferite, pentru a reduce traficul în reţea. Problema este că URL-rile nu oferă nici o posibilitate de indicare a unei pagini fără să se specifice unde este localizată pagina respectivă. Nu există nici o metodă pentru a spune ceva de genul: „Vreau pagina xyz, dar nu mă interesează de unde o aduci”. Pentru a rezolva această problemă şi a permite multiplicarea paginilor, IETF lucrează la un sistem de **URN** (**Universal Resource Names –** nume universale de resurse). Un URN poate să fie privit ca un URL generalizat. Acest subiect este în curs de cercetare, deşi o propunere de sintaxă este dată în RFC 2141.

Lipsa stării şi utilizarea cookies

Aşa cum am văzut în mod repetat, Web-ul este, în principiu, lipsit de stare. Nu există conceptul unei sesiuni de conectare. Programul de navigare transmite o cerere către server şi primeşte un fişier. Apoi serverul uită că a discutat vreodată cu acel client.

La început, când Web-ul a fost folosit doar pentru obţinerea de documente accesibile publicului larg, acest model era perfect adaptat cerinţelor. Dar, pe măsură ce Web-ul a început să capete şi alte funcţii, acest model a dat naştere unor probleme. De exemplu, anumite situri de Web impun clienţilor să se înregistreze (şi chiar să plătească bani) spre a le utiliza. Ca atare, se pune întrebarea cum pot serverele să distingă între cereri din partea utilizatorilor înregistraţi şi a celorlalţi. Un al doilea exemplu este comerţul electronic. Dacă un utilizator se plimbă printr-un magazin electronic, aruncând din când în când produse în coşul de cumpărături, cum poate serverul să reţină conţinutul coşului? Un al treilea exemplu sunt portalurile de Web configurabile cum este Yahoo. Utilizatorii pot configura o pagină iniţială detaliată, doar cu informaţia pe care o doresc (de ex.: valorile acţiunilor la bursă şi echipele lor sportive favorite), dar cum poate serverul sa afişeze pagina corectă dacă nu ştie cine este utilizatorul?

La o primă vedere, se poate crede că serverele pot să urmărească utilizatorii uitându-se la adresele lor IP. Această idee nu funcţionează însă. În primul rând, mulţi utilizatori lucrează pe calculatoare partajate cu alţi utilizatori, în special în cadrul companiilor, iar adresa IP identifică doar calculatorul nu şi utilizatorul. În al doilea rând, mult mai grav, mulţi dintre cei ce oferă servicii de Internet (ISP) utilizează NAT, astfel încât toate pachetele ce pleacă de la orice utilizator folosesc aceeaşi adresă IP. Din punctul de vedere al serverului, cele câteva de mii de clienţi ai unui ISP folosesc aceeaşi adresă IP.

Pentru a rezolva această problemă, Netscape a proiectat o tehnică mult criticată numită **cookies** *(rom. fursecuri).* Numele derivă dintr-un argou foarte vechi al programatorilor în care un program apelează o procedură şi obţine rezultate ce ar putea fi mai târziu pentru a executa ceva. În acest sens, o înregistrare de descriere a unui fişier UNIX sau un identificator al unui obiect Windows reprezintă un cookie. Mecanismul a fost formalizat mai târziu în RFC 2109.

Când un client cere o pagină de Web, serverul poate oferi informaţii adiţionale odată cu pagina cerută. Aceste informaţii pot include un cookie, care este un fişier (sau şir de caractere) de dimensiune mică (cel mult 4 KB). Programele de navigare stochează cookie-urile oferite într-un director special pentru acestea pe discul

clientului, cu excepţia cazurilor când utilizatorul a oprit utilizarea cookie-urilor. Cookie-urile sunt doar fişiere sau şiruri de caractere, nu programe executabile. În principiu, un cookie ar putea conţine un virus, dar deoarece cookie-urile sunt tratate ca date, nu există nici o posibilitate oficială ca virusul să fie executat şi să cauzeze probleme. Este însă posibil ca un hacker să exploateze o eroare a programului de navigare şi să cauzeze activarea virusului.

Un cookie poate conţine până la cinci câmpuri, aşa cum se arată în fig. 7-25. Câmpul *Domeniu* spune de unde a sosit cookie-ul. Programele de navigare trebuie să verifice că serverele nu mint în legătură cu domeniul lor. Fiecare domeniu poate stoca cel mult 20 de cookie-uri pentru fiecare client. Câmpul *Cale* reprezintă o cale în structura de directoare a serverului care identifică ce parte a arborelui de fişiere de pe server poate utiliza cookie-ul respectiv. Adesea, acest câmp este */*, ceea ce înseamnă întregul arbore.

Domeniu	Cale	Conţinut	Expiră	Sigur
toms-casino.com	/	CustomerID=497793521	15-10-02 17:00	Da
joes-store.com	/	Cart1 = l-00501;1-07031;2-	11-10-02 14:22	Nu
aportal.com	/	Prefs=Stk:SUNW+ORCL;Spt:Jet	31-12-10 23:59	Nu
sneaky.com	/	UserID=3627239101	31-12-12 23:59	Nu

Fig. 7-25. Câteva exemple de cookie-uri

Câmpul *Conţinut* are forma *nume = valoare.* Atât *nume* cât şi *valoare* pot fi orice doreşte serverul. Acesta este câmpul în care se stochează conţinutul unui cookie.

Câmpul *Expiră* arată când expiră un cookie. Dacă acest câmp este absent, programul de navigare şterge cookie-ul la terminarea execuţiei. Un astfel de cookie se numeşte **cookie ne-persistent (nonpersistent cookie)**. Dacă se oferă o dată şi o oră, cookie-ul se numeşte **persistent** şi este păstrat până la expirare. Timpul de expirare se dă pentru ora Greenwich (Greenwich Mean Time). Pentru a şterge un cookie de pe discul unui client, un server retransmite cookie-ul cu timpul de expirare în trecut.

În sfârşit, câmpul *Sigur* poate indica dacă programul de navigare poate transmite cookie-ul numai unui server sigur. Acest element este utilizat pentru comerţul electronic, aplicaţii bancare şi alte aplicaţii sigure.

Am văzut cum se obţin cookie-urile, dar cum sunt ele utilizate? Chiar înainte ca un program de navigare să transmită cererea pentru o pagină către un sit Web, el verifică directorul de cookie-uri pentru a vedea dacă există vreun cookie care a fost stocat de către domeniul la care se duce cererea. În caz afirmativ, toate cookie-urile stocate de către acel domeniu sunt incluse în mesajul ce conţine cererea. Atunci când serverul le obţine, le poate interpreta în orice mod doreşte.

Să examinăm acum câteva utilizări posibile pentru cookie-uri. În fig. 7-25, primul cookie a fost stocat de către *toms-casino.com* şi este utilizat pentru a identifica utilizatorul. Când clientul se conectează săptămâna următoare pentru a arunca nişte bani pe fereastră, programul de navigare transmite cookie-ul, astfel că serverul ştie cine este clientul. Odată ce deţine numărul de identificare al clientului, serverul poate căuta datele sale într-o bază de date şi utiliza aceste informaţii pentru a construi o pagină de Web potrivită. Depinzând de obiceiurile clientului, această pagină poate reprezenta o masă de poker, o listă a curselor de cai din ziua respectivă sau o maşină de jocuri.

Cel de-al doilea cookie a venit de la *joes-store.com*. Scenariul în acest caz este că utilizatorul se plimbă prin magazin, căutând lucruri de cumpărat. Atunci când găseşte un preţ bun şi execută un clic pe produsul respectiv, serverul construieşte un cookie ce conţine numărul de bucăţi şi codul produsului şi îl transmite clientului. Pe măsură ce clientul continuă să se plimbe prin magazin, cookie-ul este întors la fiecare nouă pagină cerută. Pe măsură ce cumpărăturile se acumulează, serverul le adaugă la cookie. În figură, coşul de cumpărături conţine trei produse, ultimul dintre ele în dublu exemplar. În cele din urmă, când clientul selectează *MERGI LA CASĂ,* cookie-ul, care acum conţine lista completă de cumpărături este trimis odată cu cererea. În acest mod, serverul ştie exact ce produse au fost cumpărate.

Cel de-al treilea cookie este pentru un portal de Web. Atunci când utilizatorul selectează o legătură către portal, programul de navigare transmite cookie-ul. Acesta spune portalului să construiască o pagină ce conţine valorile acţiunilor pentru Sun Microsystems şi Oracle şi rezultatele echipei de fotbal New York Jets. Deoarece un cookie poate avea până la 4 KB, există suficient spaţiu pentru preferinţe mai detaliate în ceea ce priveşte titluri de articole din ziar, starea vremii, oferte speciale, ş.a.m.d.

Cookie-urile pot fi utilizate şi în beneficiul serverului. De exemplu, să presupunem că un server doreşte să ştie în fiecare moment câţi vizitatori a avut şi câte pagini au fost vizitate de fiecare dintre aceştia înainte de a părăsi situl. Prima cerere nu va fi însoţită de nici un cookie, astfel că serverul va transmite înapoi un cookie ce conţine *Counter=1.* Selecţii ulterioare ale paginilor acestui site vor transmite acest cookie înapoi la server. De fiecare dată, contorul este incrementat şi transmis înapoi clientului. Urmărind aceste contoare, serverul poate să vadă câte persoane renunţă după vizitarea primei pagini, câte au vizitat două pagini, ş.a.m.d.

Cookie-urile au avut şi utilizări greşite. Teoretic, cookie-urile ar trebui să ajungă doar la situl lor de origine, dar spărgătorii au exploatat numeroase erori în programele de navigare pentru a captura cookie-uri care nu le erau destinate. Deoarece numeroase situri de comerţ electronic pun numerele de cărţi de credit în cookie-uri, potenţialul pentru abuzuri este evident.

Un mod de utilizare controversat al cookie-urilor este colectarea, în secret, de informaţii privind obiceiurile de navigare pe Web ale utilizatorilor. Mecanismul funcţionează în modul următor. O companie de publicitate, să spunem Sneaky Ads. *(rom. sneaky = viclean)* contactează un număr de situri de Web importante şi pune anunţuri publicitare ale clienţilor săi pe paginile respectivelor situri, pentru care plăteşte celor ce deţin siturile o anumită sumă. În loc să ofere sitului un fişier GIF sau JPEG pentru a fi amplasat pe fiecare pagină, le oferă un URL ce trebuie adăugat la fiecare pagină. Fiecare din aceste URL-uri conţine un număr unic în partea rezervată fişierului, cum ar fi

http://www.sneaky.com/382674902342.gif

Atunci când un utilizator vizitează o pagină *P* ce conţine un asemenea anunţ publicitar, programul de navigare obţine fişierul HTML şi vede legătura către imaginea de la *www.sneaky.com*, aşa că transmite cererea pentru imagine acestui server. Serverul întoarce un fişier GIF conţinând anunţul publicitar, împreună cu un

cookie ce conţine un număr unic de identificare pentru utilizator, 36271239101 în fig. 7-25. Compania Sneaky înregistrează faptul că utilizatorul cu acest număr de înregistrare a vizitat pagina *P*. Aceasta este uşor de realizat având în vedere faptul că fişierul cerut *(382674902342.gif)* este menţionat doar în pagina *P*. Desigur că anunţul în sine poate apare pe o mie de alte pagini, dar de fiecare dată cu un nume de fişier diferit. Compania Sneaky colectează probabil doar câţiva bănuţi de la compania ce a realizat produsul de fiecare dată când transmite anunţul publicitar.

Mai târziu, când utilizatorul vizitează o altă pagină de Web care conţine unul din anunţurile publicitare ale companiei Sneaky, după ce programul de navigare a obţinut fişierul HTML de la server, vede referinţa către, să spunem, *http:llwww.sneaky.coml493654919923.gif* şi cere acest fişier. Deoarece există deja un cookie de la domeniul *sneay.com,* programul de navigare include cookie-ul companiei Sneaky ce conţine şi numărul unic de identificare al utilizatorului. Compania Sneaky ştie acum o a doua pagină vizitată de utilizator.

Cu timpul, compania Sneaky poate construi un profil complet al obiceiurilor de navigare ale utilizatorului, deşi acesta nu a efectuat nici un clic pe vreun anunţ publicitar. Desigur, acest profil nu include încă numele utilizatorului (deşi conţine adresa IP a acestuia, informaţie care poate fi suficientă pentru a deduce numele din alte baze de date). În cazul în care utilizatorul oferă vreodată numele său unui site care cooperează cu Sneaky, un profil complet ce include şi numele este acum gata de vânzare pentru oricine doreşte să-l cumpere. Vânzarea acestor informaţii poate fi suficient de profitabilă pentru ca Sneaky sa poată plasa mai multe anunţuri publicitare pe mai multe situri Web şi astfel să colecteze şi mai multe informaţii. Cea mai ascunsă parte a acestei poveşti este că majoritatea utilizatorilor nu ştiu nimic despre această colectare de informaţii şi chiar ar putea să se creadă în siguranţă, pentru că nu au selectat nici un anunţ publicitar.

Şi dacă Sneaky vrea să fie şi mai vicleană, anunţul publicitar poate să nu fie unul clasic. Un „anunţ" format dintr-un singur pixel de culoarea fondului (şi deci invizibil) are exact acelaşi efect ca şi anunţul propriu-zis: programul de navigare trebuie să obţină imaginea gif de 1 x 1 pixeli şi să îi livreze toate cookie-urile care au fost transmise de domeniul de origine al pixelului.

Pentru a menţine impresia de intimitate, o serie de utilizatori îşi configurează programele de navigare pentru a refuza toate cookie-urile. Această acţiune poate duce însă la probleme cu siturile de Web legitime ce folosesc cookie-uri. Pentru a rezolva această problemă, utilizatorii instalează câteodată software care „mănâncă cookie-uri" *(cookie-eating software).* Acestea sunt programe speciale care inspectează fiecare cookie la sosire şi îl acceptă sau refuză în funcţie de opţiunile utilizatorului (de ex.: în ce situri Web se poate avea încredere). Această metodă oferă utilizatorului un control fin asupra căror cookie-uri sunt acceptate şi care sunt refuzate. Programele de navigare moderne cum ar fi Mozilla *(www.mozilla.org)* au un nivel de control complex asupra cookie-urilor.

7.3.2 Documente Web statice

Fundamentul Web-ului este transferul de pagini de Web de la server la client. În

forma cea mai simplă, paginile de Web sunt statice, adică doar fişiere existente pe un server, ce aşteaptă să fie cerute. În acest sens, chiar şi o înregistrare video este o pagină de Web statică, deoarece este doar un fişier. În această secţiune vom privi paginile de Web statice în detaliu. În secţiunea următoare vom examina conţinutul dinamic.

HTML--HyperText Markup Language

Paginile de Web sunt în prezent scrise într-un limbaj numit **HTML (HyperText Markup Language)**. HTML permite utilizatorilor să producă pagini de Web care conţin text, imagini şi referinţe la alte pagini de Web. HTML este un limbaj de marcare, un limbaj care descrie cum trebuie să fie formatate textele. Termenul de „marcare" provine din timpurile vechi, când editorii făceau marcaje pe documente pentru a indica tipografului – în acele timpuri un om – ce font-uri să folosească ş.a.m.d. Limbajele de marcare conţin comenzi explicite pentru formatare. De exemplu, în HTML, **înseamnă început de mod aldin, şi** înseamnă terminarea utilizării modului aldin. Avantajul utilizării unui limbaj de marcare faţă de unul în care nu se utilizează marcarea explicită constă din faptul că este simplu de scris un program de navigare care să interpreteze comenzile de marcare. TeX şi troff sunt două exemple foarte cunoscute de limbaje de marcare.

Prin standardizarea şi includerea comenzilor de marcaj în fiecare fişier HTML, devine posibil ca orice program de navigare să poată să citească şi să formateze orice pagină Web. Posibilitatea formatării paginii recepţionate este foarte importantă, deoarece o pagină poate să fie construită pe un ecran cu 1600 x 1200 pixeli utilizând culori codificate cu 24 de biţi, dar s-ar putea să fie necesară afişarea într-o mică fereastră de pe un ecran cu 640 x 320 pixeli şi utilizând culori codificate pe 8 biţi.

În cele ce urmează se face o scurtă introducere în limbajul HTML, doar pentru a oferi o idee despre subiect. Cu toate că este posibil să se construiască documente HTML utilizând orice editor, şi mulţi fac asta, este posibil şi să se utilizeze editoare HTML speciale care pot să facă toată munca (desigur, în mod corespunzător utilizatorul are mai puţin control asupra detaliilor produsului final).

O pagină Web corect formată conţine o zonă de cap şi o zonă de corp, cuprinse între marcajele (tag-uri) <html> şi </html>, dar majoritatea programelor de navigare ignoră absenţa acestor marcaje. Aşa cum se vede în fig. 7-26(a), capul este cuprins între marcajele <head> şi </head>, iar corpul între marcajele <body> şi </body>. Comenzile cuprinse între aceste marcaje se numesc **directive**. Majoritatea marcajelor HTML au acest format, adică, <ceva> pentru a indica începutul a ceva şi </ceva> pentru a marca sfârşitul. Numeroase exemple de fişiere HTML sunt disponibile. Majoritatea programelor de navigare au o opţiune VIEW SOURCE (afişarea sursei) sau ceva similar. Selectarea acestei opţiuni afişează pagina curentă în format HTML în loc de forma interpretată.

Marcajele pot să fie scrise cu litere mici sau mari. Adică <head> şi <HEAD> înseamnă acelaşi lucru, dar versiuni mai noi ale standardului cer existenţa doar a primei forme. Cum este dispus textul în documentul HTML este nesemnificativ. Programele de navigare ignoră spaţiile şi trecerile la rând nou, deoarece textul

trebuie să fie formatat pentru a corespunde zonei de afişare curente. Corespunzător, se pot utiliza spaţii pentru a face documentele HTML mai uşor de citit, ceva ce ar fi necesar pentru majoritatea documentelor. Liniile albe nu pot să fie utilizate pentru separarea paragrafelor, deoarece sunt pur şi simplu ignorate. Este necesară utilizarea unor marcaje explicite.

```
<html>
<head> <title> AMALGAMATED WIDGET, INC. </title></head>
<body> <h1> Welcome to AWI's Home Page </h1>
<img SRC="http://www.widget.com/images/logo.gif" ALT="AWI Logo"> <br> We are so happy
that you have chosen to visit <b> Amalgamated Widget's</b> home page. We hope <i> you
</i> will find all the information you need here. <p> Below we have links to information about
our many fine products. You can order electronically (by WWW), by telephone, or by FAX. </p>
<hr>
<h2> Product Information </h2>
<ul>
        <li> <a href="http://widget.com/products/big" > Big widgets </a>
        <li> <a href="http://widget.com/products/little" > Little widgets </a>
</ul>
<h2> Telephone Numbers </h2>
<ul>
        <li> By telephone: l-800-WIDGETS,
        <li> By fax: l-415-765-4321
</ul>
</body>
</hmtl>
```

(a)

Welcome to AWI's Home Page
We are so happy that you have chosen to visit Amalgamated Widget's home page. We hope you will find all the information you need here.
Below we have links to information about our many fine products. You can order electronically (by WWW), by telephone, or by FAX.
Product Information
Big widgets
Little widgets
Telephone numbers
1-800-WIDGETS
1-415-765-4321

(b)

Fig.7-26. (a) Un exemplu simplu de pagină de Web. (b) Pagina formatată.

Unele marcaje au parametri (care au nume), numiţi **atribute**. De exemplu:

```
<img src="abc" alt="foobar">
```

este un marcaj, <img>, având parametrul *src* cu valoarea *abc* şi parametrul *alt* cu valoarea *foobar.* Pentru fiecare marcaj, standardul HTML oferă o listă a parametrilor care pot să fie utilizaţi, dacă este cazul şi care este semnificaţia lor. Deoarece parametrii au nume, ordinea în care se dau valorile parametrilor nu este semnificativă.

Din punct de vedere tehnic, documentele HTML sunt scrise utilizând setul de caractere ISO 8859-l Latin-l, dar pentru utilizatorii ale căror tastaturi suportă numai codul ASCII, se pot utiliza secvenţe de caractere pentru reprezentarea caracterelor speciale cum ar fi e. Lista caracterelor speciale este precizată în standard. Toate încep cu caracterul „&” şi se termină cu „;”. De exemplu è produce e iar é produce e. Deoarece <, > şi & au semnificaţii speciale, pot să fie reprezentate numai utilizând secvenţele speciale de caractere corespunzătoare, < > şi &.

Principalul element din zona de cap este titlul care este cuprins între <title> şi </title>, dar aici pot să apară şi alte tipuri de informaţii. Titlul nu este afişat pe pagină. Unele programe de navigare îl utilizează pentru a eticheta fereastra în care se afişează pagina respectivă.

Să analizăm şi alte particularităţi prezente în exemplul din fig. 7-26. Toate marcajele utilizate în fig. 7-26 şi încă alte câteva sunt prezentate în fig. 7-27. Titlurile de capitole sunt generate de marcajul <h*n*>, unde *n* este o cifră între 1 şi 6. <h1> este titlul cel mai important; <h6> este cel mai puţin important.

Marcaj	Descriere
<html> ... </html>	Delimitează textul scris în HTML
<head> ... </head>	Delimitează zona de cap
<title> ... </title>	Defineşte titlul (nu este afişat de programul de
<body> ... </body>	Delimitează zona de corp
<h*n*> ... </h*n*>	Delimitează un titlu de nivel *n*
...	Textul ... o să fie afişat cu aldine
...	Textul ... o să fie afişat cu cursive
...	Centrează ... pe pagină orizontal
...	Delimitează o listă neordonată
...	Delimitează o listă ordonată (numerotată)
...	Delimitează un elemente într-o listă ordonată sau neordonată
 	Trecere la linie nouă
<P>	Început de paragraf
<hr>	Linie orizontală
<img src=„...”>	Se încarcă o imagine
 ...	Se defineşte o hiper-legătură

Fig. 7-27. O selecţie de marcaje uzuale. Unele mai au şi alţi parametri.

Depinde de programul de navigare să prezinte aceste titluri în mod diferit pe ecran. De obicei, titlurile cu număr mai mic vor fi afişate utilizând caractere mai mari. Programul de navigare poate să utilizeze culori diferite pentru fiecare nivel de titlu. De obicei, pentru titlurile marcate cu <h1 > se utilizează litere mari scrise cu aldine cu cel puţin o linie liberă înainte şi după. Corespunzător, pentru titlurile

marcate cu <h2>, se utilizează caractere mai mici cu mai puţin spaţiu lăsat înainte şi după.

Marcajele <b> şi <i> sunt utilizate pentru a indica modurile aldin şi respectiv cursiv. Dacă programul de navigare nu poate să afişeze aceste tipuri de caractere, va utiliza un alt mod de a le reprezenta, de exemplu utilizând diferite culori sau video-invers.

Limbajul HTML oferă diferite mecanisme pentru construirea de liste, inclusiv liste conţinute în alte liste. Listele încep cu <ul> sau <ol>, <li> fiind folosit pentru a marca începutul elementelor în ambele cazuri. Marcajul <ul> indică începutul unei liste neordonate. Elementele individuale, care sunt marcate în sursă cu <li>, sunt reprezentate precedate de buline (•). O variantă a acestui mecanism este <ol>, care descrie o listă ordonată. Când se utilizează acest marcaj, textele precedate de <li> sunt numerotate de către programul de navigare. Marcajele <ul> şi <ol> au aceeaşi sintaxă şi efecte asemănătoare.

Marcajele <br>, şi <hr> indică o separare între diferitele părţi ale textului. Formatul precis este descris în pagina de stil (vezi mai jos) asociată cu pagina curentă. Marcajul <br> forţează trecerea la linie nouă. De obicei programele de navigare nu inserează o linie liberă după <br>. Marcajul reprezintă un început de paragraf, pentru care se va insera o linie nouă şi eventual se va face o indentare. (În mod teoretic, există şi pentru a indica sfârşitul de paragraf, dar este foarte rar utilizat; majoritatea celor care scriu în HTML nici nu ştiu că acest marcaj există). Ultimul marcaj, <hr>, forţează trecerea la linie nouă şi desenează o linie orizontală.

HTML permite includerea de imagini în paginile de Web. Marcajul <img> arată că pe poziţia curentă din pagină se va include o imagine. Marcajul poate să aibă o serie de parametri. Parametrul *src* indică URL-ul imaginii. Standardul HTML nu specifică ce formate grafice sunt permise. În practică, toate programele de navigare acceptă fişiere în format GIF, multe pot lucra şi cu fişiere în format JPEG. Programele de navigare pot să lucreze şi cu alte formate, dar o astfel de extensie este o sabie cu două tăişuri. Dacă, de exemplu un utilizator este obişnuit cu un program de navigare care suportă fişiere în format BMP, este posibil ca el să includă astfel de fişiere în pagina sa de Web şi să fie uimit că alte programe de navigare ignoră imaginile sale minunate.

Alţi parametri pentru <img> sunt *align*, care controlează modul în care se aliniază imaginea faţă de limita de jos a textului (*top, middle, bottom*), *alt*, care furnizează textul afişat în locul imaginii dacă utilizatorul dezactivează opţiunea de afişare a imaginilor şi *ismap*, un indicator care anunţă că imaginea este o hartă selectabilă.

În sfârşit, să considerăm hiper-legăturile, care utilizează marcajele <a> (anchor) şi . Ca şi <img>, <a> are diverşi parametrii, printre care *href* (URL-ul) şi *name* (numele hiper-legăturii). Textul cuprins între <a> şi este afişat. Dacă este selectat, atunci se utilizează hiper-legătura pentru a se aduce o nouă pagină. În locul textului se poate pune şi un marcaj <img>, caz în care, cu un clic pe imagine, se va activa legătura.

De exemplu, să considerăm următorul fragment HTML:

```
<a href=„http://www.nasa.gov">NASA's home page </a>
```

Când se afişează acest fragment, pe ecran apare:

NASA's home page

Dacă utilizatorul execută un clic pe acest text, programul de navigare aduce şi afişează pagina al cărei URL este *http://www.nasa.gov*.

Ca al doilea exemplu, să considerăm

```
<a href=„http://www.nasa.gov"><img src=„shuttle.gif"alt=„NASA"></a>
```

Când se afişează pagina va apărea o imagine (o navetă spaţială). Executând un clic pe imagine, se va aduce pagina NASA, la fel ca şi în cazul în care în exemplul anterior a fost selectat textul. Dacă utilizatorul a dezactivat opţiunea de afişare a imaginilor, atunci în loc de imagine va fi afişat textul NASA.

```
<html>
<head><title> A sample page with a table </title></head>
<body>
<table border=1 rules=all>
<caption> Some differences between HTML Versions </caption>
<col align=left>
<col align=center>
<col align= center >
<col align= center >
<col align= center >
<tr> <th>Item <th>HTML 1.0 <th>HTML 2.0 <th>HTML 3.0 <th> HTML 4.0 </tr>
<tr> <th> Hyperlinks <td> x <td> x <td> x <td> x </tr>
<tr> <th> Images <td> x <td> x <td> x <td> x </tr>
<tr> <th> Lists <td> x <td> x <td> x <td> x </tr>
<tr> <th> Active Maps and Images <td>   <td> x <td> x <td> x </tr>
<tr> <th> Forms <td>   <td> x <td> x <td> x </tr>
<tr> <th> Equations <td>   <td>   <td> x <td> x </tr>
<tr> <th> Toolbars <td>   <td>   <td> x <td> x </tr>
<tr> <th> Tables <td>   <td>   <td> x <td> x </tr>
<tr> <th> Accesibility features <td>   <td>   <td>   <td> x </tr> <tr>
<th> Object embedding <td>   <td>   <td>   <td> x </tr> <tr> <th>
Scripting <td>   <td>   <td>   <td> x </tr>
</table>
</body>
</html>
```

(a)

Some diferences between HTML Versions

Item	HTML 1.0	HTML 2.0	HTML 3.0	HTML 4.0
Hyperlinks	x	x	x	x
Images	x	x	x	x
Lists	x	x	x	x
Active Maps and Images		x	x	x
Forms		x	x	x
Equations			x	x

Toolbars			x	x
Tables			x	x
Accessibility features				x
Object embedding				x
Scripting				x

(b)

Fig. 7-28. (a) O tabelă HTML, (b) Un rezultat posibil.

Pentru marcajul <a> se poate utiliza parametrul *name* pentru a fixa o hiper-legătură, care să fie referită din pagină. De exemplu, unele pagini de Web încep cu o tabelă de conţinut selectabilă. Prin execuţia unui clic pe o intrare în tabela de conţinut, se va trece direct la secţiunea corespunzătoare din pagină.

HTML evoluează continuu. În versiunile HTML 1.0 şi HTML 2.0 nu existau tabele, dar au fost adăugate în HTML 3.0. O tabelă HTML este formată din una sau mai multe linii, fiecare fiind formată din una sau mai multe **celule**. Celulele pot să conţină diferite tipuri de informaţii, inclusiv text, figuri, iconiţe, fotografii şi chiar tabele. Celulele pot să fie alipite, de exemplu un titlu poate să se întindă peste mai multe coloane. Autorii paginilor au control asupra modului în care se face afişarea, inclusiv alinierea, stilul bordurii, marginile celulelor, dar programul de navigare este cel care hotărăşte de fapt cum se face afişarea.

O descriere în HTML a unei tabele este prezentată în fig. 7-28(a), iar efectul posibil este prezentat în fig. 7-28(b). Acest exemplu prezintă câteva din facilităţile de descriere a tabelelor în HTML. Tabelele încep cu marcajul <table>. Se pot specifica informaţii suplimentare pentru a descrie proprietăţile generale ale tabelei.

Marcajul <caption> poate să fie utilizat pentru a furniza un titlu tabelei. Fiecare linie începe cu marcajul <tr> (Table Row – linie în tabelă). Celulele individuale sunt marcate cu <th> (Table Header – titlu de coloană), <td> (Table Data – date în tabelă). Diferenţierea este necesară pentru a permite programului de navigare să le afişeze diferit, aşa cum se vede şi din exemplul considerat.

În tabele se pot utiliza alte marcaje. Acestea includ posibilitatea de a specifica alinieri orizontale sau verticale ale celulelor, alinierea în cadrul celulei, margini, gruparea de celule, unităţi şi multe altele.

În HTML 4.0 au fost adăugate noi elemente. Acestea includ elemente ce fac paginile mai accesibile utilizatorilor cu handicap, înglobarea obiectelor (o generalizare a marcajului <img> astfel încât alte obiecte să poată fi înglobate în pagini), suport pentru limbaje de scripturi (pentru a permite conţinut dinamic) şi multe altele.

Când un sit de Web este complex, fiind format din multe pagini produse de autori diferiţi ce lucrează pentru aceeaşi companie, este adesea de dorit să existe o modalitate pentru a împiedica moduri de prezentare diferite în pagini diferite. Această problemă poate fi rezolvată utilizând **paginile de stil (style sheets)**. Atunci când se utilizează pagini de stil, paginile individuale nu mai folosesc stiluri fizice, cum sunt modurile aldin şi cursiv. Autorii pot acum să utilizeze stiluri logice, cum sunt <dn> (definiţie), (evidenţiere), (evidenţiere accentuată) şi

<var> (variabile de program). Stilurile logice sunt definite în pagina de stil, pentru care există o referinţă la începutul fiecărei pagini. În acest fel, toate paginile au acelaşi stil şi dacă administratorul sitului *(Webmaster)* decide să schimbe stilul din stil cursiv de 14 puncte tipografice, culoare albastră în stil aldin, 18 puncte tipografice, culoare roz ţipător, tot ceea ce trebuie să facă este să schimbe o singură definiţie pentru a converti întregul sit Web. O pagină de stil poate fi comparată cu o directivă #în- clude într-un program C: schimbarea unei macrodefiniţii în fişierul inclus determină schimbarea în toate fişierele program ce includ respectivul header.

Formulare

HTML 1.0 funcţiona într-o singură direcţie. Utilizatorii puteau să aducă o pagină de la furnizorii de informaţie, dar era foarte dificil să se transmită informaţie în sens invers. Pe măsură ce tot mai multe organizaţii comerciale au început să utilizeze Web-ul, a apărut o puternică cerere pentru comunicaţia în dublu sens. De exemplu, multe companii vor să poată prelua comenzi pentru produse utilizând paginile lor de Web, furnizorii de software vor să distribuie programe prin intermediul Web-ului, clienţii să îşi completeze fişele de înregistrare prin acelaşi mijloc, iar companiile care oferă servicii de căutare în Web au nevoie ca utilizatorii de servicii să poată să introducă cuvintele pe baza cărora se face căutarea.

Acest gen de cereri a dus la includerea **formularelor** începând cu HTML 2.0. Formularele conţin casete şi butoane care permit utilizatorilor să completeze informaţii sau să facă selecţii şi apoi să transmită informaţiile la proprietarul paginii. În acest scop se utilizează marcajul <input>. Acesta are o varietate de parametri care determină mărimea, tipul, şi modul de afişare a casetei utilizate. Cele mai obişnuite sunt câmpuri în care utilizatorul poate să introducă text, casete care pot să fie selectate, hărţi active, butoane *submit.* Exemplul din fig. 7-29 prezintă câteva dintre aceste posibilităţi.

```
<html>
<head><title> AWI CUSTOMER ORDERING FORM </title></head> <body>
<h1> Widget Order Form </h1>
<form ACTION="http://widget.com/cgi-bin/widgetorder" method=POST>
<p> Name <input name="customer" size=46> </p>
<p> Street Address <input name="address" size=40> </p>
<p> City <input name="city" size=20> State <input name="state" size=4>
Country <input name="country" size=10> </p>
<p> Credit card # <input name="cardno" size=10> expires <input name="expires" size=4>
M/C <input name="cc" type=radio value="mastercard"> VISA <input name="cc" type=radio
value="visacard"> </p> <p> Widget size Big <input name="product" type=radio
value="expensive"> Little <input name="product" type=radio value="cheap">
Ship by express courier <input name="express" type=checkbox> </p>
<p> <input type=submit value="submit order"> </p> Thank you for ordering an AWI widget,
the best widget money can buy! </form> </body> </html>
```

(a)

Widget Order Form

Name

Street address

City State CountryP

Credit card # Expires M/C Q Visa O

Widget size Big Q Little Q Ship by express courier | |

Submit order

Thank you for ordering an AWI widget, the best widget money can buy!

(b)

Să începem discuţia parcurgând exemplul. Ca orice formular, şi acesta este cuprins între marcajele <form> şi </form>. Textele care nu sunt incluse între marcaje sunt afişate. Într-un formular poate să fie utilizat orice marcaj obişnuit (de

Fig. 7-29. (a) Un formular de comandă HTML. (b) Pagina formatată.

exemplu <b>) În acest formular sunt utilizate trei tipuri de casete.

Prima casetă din formular apare după textul „Name”. Caseta are lăţimea de 46 de caractere şi utilizatorul va introduce un şir care va fi memorat în variabila *customer* pentru prelucrări ulterioare. Marcajul <p> indică programului de navigare să afişeze ceea ce urmează pe o linie nouă, chiar dacă mai este loc pe linia curentă. Utilizând <p> şi alte marcaje care controlează dispunerea textului, creatorul paginii poate să controleze cum arată formularul pe ecran.

Pe linia următoare se solicită adresa utilizatorului, având cel mult 40 de caractere, pe o linie separată. Urmează o linie pe care se solicită oraşul, statul şi ţara. Aici nu se utilizează marcajul <p>, deci programul de navigare o să le afişeze pe toate pe aceeaşi linie, dacă încap. Din punctul de vedere al programului de navigare, paragraful curent conţine şase elemente: trei şiruri alternând cu trei casete. El le afişează pe aceeaşi linie de la stânga la dreapta, trecând la o linie nouă ori de câte ori pe linia curentă nu mai încape următorul element. Astfel, este posibil ca pe un ecran 1600 x 1200 să încapă toate cele trei şiruri şi casetele corespunzătoare, în timp ce pe un ecran 1024 x 768 ele pot să fie distribuite în două linii. În cazul cel mai defavorabil cuvântul „Country” este la capătul liniei, iar caseta asociată este la începutul liniei următoare. Nu există nici o posibilitate de a forţa programul de navigare să afişeze caseta lângă text.

Următoarea linie solicită numărul cărţii de credit şi data sa de expirare. Transmiterea numerelor cărţilor de credit prin Internet trebuie să se facă numai dacă s-au luat măsurile de securitate adecvate. Vom discuta despre aceste măsuri în cap. 8.

După data de expirare, întâlnim un element nou: butoane radio. Acestea sunt utilizate atunci când trebuie să se facă o alegere între mai multe alternative. Modelul care se utilizează aici este cel al unui aparat de radio care are butoane pentru selecţia scalelor. Programul de navigare afişează aceste casete într-o formă care permite utilizatorului să le selecteze sau să le deselecteze prin execuţia unui clic (sau utilizând tastatura). Selecţia unuia dintre ele le deselectează pe toate celelalte care fac parte din acelaşi grup. Modul de afişare depinde de programul de navigare. „Widget size” utilizează de asemenea două butoane. Cele două grupuri sunt diferenţiate prin câmpul *name* şi nu printr-un domeniu static de genul <radiobutton> ... </radiobutton>.

Parametrii *value* sunt utilizaţi pentru a arăta care buton a fost apăsat. În funcţie de opţiunea aleasă pentru cartea de credit, variabila *cc* va avea ca valoare şirul „mastercard” sau „visacard”.

După cele două seturi de butoane, urmează opţiunea referitoare la modul de transport, reprezentată de o casetă de tip *checkbox.* Aceasta poate să fie pe poziţia selectat sau nu. Spre deosebire de cazul butoanelor radio, unde poate să fie selectat un singur buton dintr-un set, fiecare casetă de tip *checkbox* poate să fie selectată sau nu, independent de celelalte. De exemplu, când se comandă o piţa utilizând pagina de Web Electropizza, utilizatorul poate să aleagă sardele *şi* ceapă *şi* ananas (dacă le suportă), dar nu poate să aleagă mică *şi* medie *şi* mare pentru aceeaşi piţa. Conţinutul corespunzător piţei va fi reprezentat de trei butoane diferite de tip

checkbox, în timp ce dimensiunea va fi reprezentată de un set de butoane radio.

Pe de altă parte, în cazul în care lista din care se face alegerea este foarte lungă, butoanele radio devin dificil de utilizat. Din acest motiv, marcajele <select> şi </select> sunt utilizate pentru a prezenta o listă de alternative, utilizând semantica corespunzătoare unor butoane radio (dacă nu se utilizează parametrul *multiple,* caz în care semantica este cea de la casetele de tip *checkbox).* Unele programe de navigare afişează opţiunile cuprinse între <select> şi </select> ca un meniu derulant.

Am văzut jumătate din tipurile standard pentru marcajul <input>: *radio* şi *checkbox.* De fapt, am văzut şi un al treilea: *text.* Deoarece acesta este tipul implicit, nu am mai utilizat parametrul *type = text*, dar puteam să o facem. Alte două tipuri sunt *password* şi *textarea.* O casetă *password* funcţionează la fel ca o casetă *text,* numai că nu se face afişarea caracterelor introduse. O casetă *textarea* este similară unei casete *text*, numai că va conţine mai multe linii.

Întorcându-ne la exemplul din fig. 7-29, urmează butonul *submit.* Când este selectat acest buton, informaţia introdusă de către utilizator este transmisă la calculatorul de pe care provine formularul. Similar altor tipuri, *submit* este un cuvânt cheie pe care îl înţelege programul de navigare. Şirul *value* reprezintă în acest caz eticheta butonului şi se afişează. Toate casetele pot să aibă valori, dar am avut nevoie de această facilitate numai aici. Pentru casetele *text*, conţinutul câmpului *value* este afişat o dată cu formularul, dar utilizatorul poate să îl modifice sau să îl şteargă. Casetele *checkbox* şi *radio* pot să fie iniţializate, utilizând însă un parametru numit *checked* (deoarece parametrul *value* oferă un text, dar nu indică o selecţie).

Atunci când utilizatorul selectează butonul „submit order”, programul de navigare împachetează informaţia colectată într-o singură linie lungă şi o transmite serverului pentru procesare. Pentru a separa câmpurile, se utilizează caracterul &; caracterul + reprezintă spaţiu. De exemplu, răspunsul la formular poate să arate ca în fig. 7-30:

```
(împărţit aici în trei linii din motive de aliniere pagină)
customer=John+Doe&address=100+Main+St.&city=White+Plain&
state=NY&country=USA&cardno=1234567890&expires6/98&cc=mastercard&
product=cheap&express=on
```

Fig. 7-30. Un răspuns posibil de la programul de navigare către server cu informaţiile completate de utilizator

Şirul va fi transmis la server ca o singură linie, nu ca trei. Dacă un *checkbox* nu este selectat, el este omis din şir. Este problema serverului să interpreteze şirul respectiv. Vom discuta cum se poate realiza acest lucru mai târziu în acest capitol.

XML şi XSL

Limbajul HTML, cu sau fără formulare, nu conferă nici o structură paginilor de

Web. De asemenea, el amestecă conţinutul cu informaţii despre formatul paginii. Pe măsură ce comerţul electronic şi alte aplicaţii devin din ce în ce mai răspândite, apare o cerere din ce în ce mai mare pentru structurarea paginilor de Web şi separarea conţinutului de informaţiile de format. De exemplu, un program care caută pe Web preţul cel mai bun al unei cărţi sau al unui CD trebuie să analizeze multe pagini de Web căutând numele produsului şi preţul său. Cu paginile de Web în format HTML este foarte dificil ca un program să îşi dea seama unde se află numele şi unde se află preţul.

Din acest motiv, consorţiul W3C a dezvoltat îmbunătăţiri ale limbajului HTML pentru a permite paginilor de Web să fie structurare în vederea procesării automate. Două limbaje noi au fost dezvoltate în acest scop. Mai întâi, **XML (eXtensible Markup Language)** descrie conţinutul într-un mod structurat şi apoi, **XSL (eXtensible Style Language)** descrie formatul independent de conţinut. Ambele limbaje reprezintă subiecte mari şi complexe, ca atare scurta introducere care urmează atinge tangenţial subiectul, deşi ar trebui să ofere o idee despre modul cum funcţionează.

Să considerăm documentul XML dat ca exemplu în fig. 7-31. Acesta defineşte o structură numită book_list, care reprezintă o listă de cărţi. Fiecare carte (book) are trei câmpuri, titlul, autorul şi anul publicării. Aceste structuri sunt extrem de simple. Sunt permise structurile ce conţin câmpuri repetitive (de ex.: mai mulţi autori), câmpuri opţionale (de ex.: titlul CD-ROM-ului inclus) şi câmpuri la alegere (de ex.: URL-ul unei librării dacă respectiva carte mai este disponibilă sau URL-ul unui sit de licitaţii dacă nu mai este disponibilă pe piaţă).

În acest exemplu, fiecare din cele trei câmpuri este o entitate indivizibilă, dar se permite subdivizarea ulterioară a unui câmp. De exemplu, câmpul autor putea fi alcătuit aşa cum urmează, spre a oferi un control mai fin la căutare şi formatare:

```
<author>
        <f i rst_n a m e>An d rew</f i rst_n ame>
        <last_name>Tannenbaum</last_name>
</author>
```

Fiecare câmp poate fi împărţit în sub-câmpuri şi sub-sub-câmpuri până la o adâncime arbitrară.

Întregul fişier din fig. 7-31 defineşte o listă de cărţi ce conţine trei cărţi. Nu spune nimic despre modul cum se poate afişa pagina de Web pe ecran. Pentru a oferi informaţii de formatare avem nevoie de un al doilea fişier, *book_list.xsl,* ce conţine definiţia XSL. Acest fişier este o pagină de stil care spune cum se poate afişa pagina. (Există alternative la paginile de stil, cum ar fi conversia XML în HTML, dar aceste alternative depăşesc subiectul acestei cărţi.)

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href="book_list.xsl"?>

<book_list>

<book>
        <title>Computer Networks, 4/e </title>
        <author> Andrew S. Tannenbaum </author>
```

```
        <year> 2003 </year>
</book>

<book>
        <title> Modern Operating Systems, 2/e </title>
        <author> Andrew S. Tannenbaum </author>
        <year> 2001 </year>
</book>

<book>
        <title> Structured Computer Organization 4/e </title>
        <author> Andrew S. Tannenbaum </author>
        <year> 1999 </year>
</book>

</book_list>
```

Fig. 7-31. O pagină de Web simplă în format XML

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version=" 1.0">
<xsl:template match="/">

<html> <body> <table border="2"> <tr>

<th> Title </th>
<th> Author </th>
<th> Year </th>
</tr>

<xsl:for-each select="book_list/book">
<tr>
<td> <xsl: value-of select="title"/> </td>
<td> <xsl: value-of select="author"/> </td> <td> <xsl: value-of select="year"/> </td>
</tr>
</xsl:for-each>
</table>

</body> </html> </xsl:template> </xsl:stylesheet>
```

Fig. 7-32. O pagină de stil în XSL

Un exemplu de fişier XSL pentru formatarea conţinutului din fig. 7-31 este dat în fig. 7-32. După câteva declaraţii necesare ce includ URL-ul standardului XSL, fişierul conţine marcaje începând cu <html> şi <body>. Acestea definesc începutul unei pagini de Web, ca de obicei. Urmează apoi o definiţie de tabel ce include titlurile celor trei coloane. Să observăm că în plus faţă de marcajele <th> există şi marcaje </th>, lucru care nu ne-a preocupat până acum. Specificaţiile XML şi XSL sunt mult mai stricte decât specificaţia HTML. Ele statuează că rejectarea fişierelor incorecte din punct de vedere sintactic este obligatorie, chiar dacă programul de navigare poate determina ce a intenţionat proiectantul paginii Web. Un program de navigare care acceptă fişiere XML sau XSL incorecte din punct de vedere sintactic şi repară eroarea el însuşi este neconform cu standardul şi va fi rejectat la un test de conformanţă cu standardele. Programelor de navigare li se permite însă să identifice exact eroarea. Această măsură întrucâtva draconică este necesară pentru a rezolva problema numărului imens de pagini de Web scrise neglijent, care există în prezent.

Instrucţiunea

```
<xsl: for-each select="book_list/book">
```

este comparabilă cu o instrucţiune *for* în C. Execuţia ei determină programul de navigare să execute corpul buclei (terminată de <xsl:for-each>) câte o dată pentru fiecare carte. Fiecare iteraţie afişează cinci linii: <tr>, titlul, autorul şi anul şi </tr>. După această buclă, sunt transmise la ieşire marcajele de închidere <body> şi </html>. Rezultatul operaţiei programului de navigare de a interpreta această pagină de stil este acelaşi ca şi în cazul când pagina de Web ar fi conţinut direct tabelul. Totuşi, în acest format, programele pot analiza fişierul XML şi găsi cu uşurinţă cărţile publicate du-

pă 2000, de exemplu. Merită subliniat faptul că deşi fişierul nostru XSL conţinea un fel de buclă, paginile de Web în XML şi XSL sunt în continuare statice deoarece conţin pur şi simplu instrucţiuni pentru programul de navigare despre modul de afişare a paginii, la fel ca şi paginile HTML. Desigur, pentru a folosi XML şi XSL, programul de navigare trebuie să fie capabil să interpreteze XML şi XSL, dar marea majoritate a acestora au deja această capacitate. Nu este încă foarte clar dacă XSL va prelua paginile de stil tradiţionale.

Nu am arătat cum se poate face acest lucru, dar limbajul XML permite proiectantului de situri Web să creeze fişiere de definiţie în care structurile sunt definite în avans. Aceste fişiere de definiţii pot fi incluse unele în altele, făcând posibilă construcţia de pagini Web complexe. Pentru informaţii suplimentare despre aceasta şi multe alte caracteristici ale limbajelor XML şi XSL, consultaţi una din multele cărţi despre acest subiect. Două exemple sunt (Livingston, 2002; şi Williamson, 2001).

Înainte de a încheia discuţia despre XML şi XSL, merită să comentăm pe marginea luptei ideologice din interiorul consorţiului WWW şi a comunităţii dezvoltatorilor de pagini Web. Scopul iniţial al limbajului HTML era să specifice *structura* documentului şi nu *modul de afişare.* De exemplu,

```
<h1> Deborah's Photos </h1>
```

instruieşte programul de navigare să sublinieze titlul, dar nu spune nimic despre tipului font-ului, dimensiune sau culoare. Acestea sunt lăsate pe seama programului de navigare, care cunoaşte proprietăţile ecranului (de ex.: câţi pixeli are). Totuşi, mulţi dezvoltatori de pagini Web doreau controlul absolut asupra modalităţii în care erau afişate paginile lor, astfel că au fost adăugate noi marcaje la HTML pentru a controla modul de afişare, cum ar fi

```
<font face="helvetica" size="24" color="red"> Deborah's Photos </font>
```

De asemenea, au fost adăugate modalităţi de a controla poziţionarea precisă pe ecran. Această abordare este că nu este portabilă, ceea ce reprezintă desigur o problemă. Deşi o pagină poate fi afişată perfect de programul de navigare cu care este dezvoltată, un alt program de navigare, sau chiar altă versiune a aceluiaşi program sau o altă rezoluţie a ecranului poate fi un dezastru. XML este parţial o încercare de întoarcere la scopul originar de a specifica doar structura nu modalitatea de afişare a documentului. Totuşi, XSL este oferit în plus, pentru a controla modul de afişare. Ambele formate pot avea însă utilizări eronate. Puteţi conta pe asta.

XML poate fi utilizat şi în alte scopuri decât acela de a descrie pagini de Web. O utilizare din ce în ce mai frecventă este aceea de limbaj de comunicare între aplicaţii. În particular, **SOAP (Simple Object Access Protocol –** Protocol simplu pentru accesul la obiecte**)** este o modalitate de a executa apeluri de tip RPC între aplicaţii într-un mod independent de limbaj şi de sistem. Clientul construieşte cererea ca mesaj XML şi o transmite serverului, utilizând protocolul HTTP (descris mai departe). Serverul trimite înapoi un răspuns sub formă de mesaj XML. În acest mod pot comunica aplicaţii de pe sisteme heterogene.

XHTML – eXtended HyperText Markup Language

Limbajul HTML continuă să evolueze pentru a se conforma noilor cereri. Multe persoane din acest domeniu cred că în viitor majoritatea dispozitivelor cu acces la Web nu vor fi calculatoarele personale, ci dispozitive cu conexiuni fără fir, de tip PDA. Aceste dispozitive au memorie limitată pentru programe de navigare mari cu metode euristice ce încearcă să trateze într-un anumit mod paginile de Web incorecte din punct de vedere sintactic. Astfel, următorul pas după HTML 4 este un limbaj care este Foarte Selectiv. Acest limbaj este numit **XHTML (eXtended HyperText Markup Language, rom.: Limbaj extins de marcaje hipertext)** mai degrabă decât HTML 5, pentru că, de fapt, este HTML 4 reformulat în XML. Prin aceasta vrem să spunem că marcaje precum <h1 > nu au nici o însemnătate prin ele însele. Pentru a obţine efectul din HTML 4 este nevoie de o definiţie în fişierul XSL. XHTML este noul standard pentru Web şi ar trebui folosit pentru toate paginile de Web noi pentru a asigura un maxim de portabilitate pe diverse platforme şi programe de navigare.

Există şase diferenţe majore şi mai multe diferenţe minore între XHTML şi HTML 4. Să trecem acum în revistă diferenţele majore. Mai întâi, paginile XHTML şi programele de navigare trebuie să se supună în mod strict standardului. Gata cu paginile de Web de proastă calitate. Această proprietate a fost moştenită din XML.

În al doilea rând, toate marcajele şi atributele trebuie să fie scrise cu litere mici. Marcaje ca <HTML> nu sunt valide în XHTML. Folosirea marcajelor precum <html> este acum obligatorie. Similar, <img SRC=”pic001.jpg”> este interzis pentru că are în componenţă un atribut scris cu litere mari.

În al treilea rând, sunt necesare marcaje de terminare, chiar şi pentru </p>. Pentru marcaje care nu au un marcaj natural de terminare, cum ar fi <br>, <hr> şi <img>, un caracter / trebuie să preceadă caracterul de terminare „>”, de exemplu

```
<img src="pic001.jpg" />
```

În al patrulea rând, atributele trebuie să fie conţinute între ghilimele. De exemplu,

```
<img SRC="pic001.jpg" height=500 />
```

nu mai este permis. Valoarea 500 trebuie pusă între ghilimele, cum este numele fişierului JPEG, chiar dacă 500 este doar un număr.

În al cincilea rând, marcajele trebuie să se conţină unul pe altul într-un mod corespunzător. În trecut, acest lucru nu era necesar atât timp cât starea finală atinsă era corectă. De exemplu,

```
<center> <b> Vacation Pictures </center> </b>
```

era legal. În XHTML nu mai este. Marcajele trebuie închise în ordinea inversă în care au fost deschise.

În al şaselea rând, fiecare document trebuie să-şi specifice tipul documentului. De exemplu, am văzut acest lucru în fig. 7-32. Pentru o discuţie asupra schimbărilor, fie ele majore sau minore, vezi *www.w3.org*.

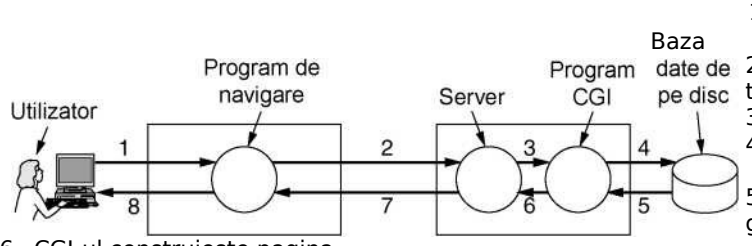
7.3.3 Documente Web dinamice

Până acum, modelul pe care l-am folosit este cel din Fig. 6-6: clientul transmite numele fişierului către server, care apoi întoarce fişierul. La începutul Web-ului, tot conţinutul era de fapt static în acest mod (doar fişiere). Totuşi, în ultimii ani, din ce în ce mai mult conţinut a devenit dinamic, adică generat la cerere şi nu doar stocat pe disc. Generarea de conţinut poate avea loc fie la server, fie la client. Să examinăm acum pe rând fiecare din aceste cazuri.

Generare dinamică de pagini de Web la server

Pentru a vedea de ce este necesară generarea de conţinut la server, să luăm în considerare utilizarea formularelor, aşa cum a fost descrisă mai devreme. Atunci când un utilizator completează un formular şi apasă butonul *submit,* se transmite un mesaj către server, mesaj ce arată că are în interior conţinutul unui formular, împreună cu acele câmpuri completate de utilizator. Acest mesaj nu este numele unui fişier ce trebuie întors. În schimb, acest mesaj trebuie să fie oferit unui program, sau

script, pentru a fi procesat. De obicei, procesarea implică folosirea informaţiilor oferite de utilizator pentru căutarea unei înregistrări într-o bază de date de pe discul serverului şi generarea unei pagini HTML personalizate pentru a fi trimisă înapoi clientului. De exemplu, într-o aplicaţie de comerţ electronic, atunci când utilizatorul face un clic pe *MERGI LA CASĂ,* programul de navigare întoarce cookie-ul ce conţine produsele din coşul de cumpărături, dar la server trebuie apelat un program, sau script, care procesează acest cookie şi generează o pagin ă HTML ca răspuns. Pagina HTML ar putea afişa un formular ce conţine lista de produse din coş şi ultima adresă de expediere cunoscută a utilizatorului, împreună cu o cerere de verificare a informaţiilor şi de specificare a modalităţii de plată. Etapele necesare pentru procesarea informaţiei dintr-un formular HTML sunt ilustrate în fig. 7-33.



1. Utilizatorul completează formularul
2. Formularul este transmis
3. Trimis CGI-ului
4. CGI-ul interoghează baza de date
5. înregistrarea este găsită
6. CGI-ul construieşte pagina
7. Pagina este întoarsă clientului
8. Pagina este afişată

Fig. 7-33. Etapele de procesare a informaţiei dintr-un formular HTML

Modalitatea tradiţională de a trata formularele şi alte pagini de Web interactive este sistemul numit **CGI** (**Common Gateway Interface –** Interfaţă comună de conversie). Aceasta este o interfaţă standardizată ce permite serverelor de Web să discute cu programele din fundal şi cu scripturile care acceptă o intrare (de ex.: formulare) şi să genereze pagini HTML ca răspuns. De obicei, aceste programe de fundal sunt scripturi scrise în limbajul Perl deoarece scripturile Perl sunt mai uşor şi mai rapid de scris decât programele (cel puţin dacă ştii să programezi în Perl). În mod obişnuit, ele sunt localizate într-un director numit *cgi-bin*, care este vizibil în URL. Câteodată un alt limbaj de scripturi, Python, este utilizat în loc de Perl.

Ca un exemplu de cât de frecvent lucrează CGI, să considerăm cazul unui produs al companiei Truly Great Products Company *(rom. Compania Produselor cu Adevărat Bune)* care vine cu o fişă de înregistrare a produsului pentru garanţie. În loc de a completa această fişă, clientului i se spune să meargă la *www.tgpc.com* pentru a se înregistra on-line. Pe acea pagină există o hiper-legătură care spune

Apăsaţi aici pentru a va înregistra produsul

Această legătura indică spre un script Perl, să spunem *www.tgpc.com/cgi-bin/reg.perl.* Când acest script este executat fără parametri, transmite înapoi o pagină

HTML conţinând formularul de înregistrare. Atunci când utilizatorul completează formularul şi face un clic pe *submit* se transmite un mesaj acestui script, mesaj ce conţine valorile completate după modelul din fig. 7-30. Scriptul Perl analizează parametrii, adaugă o înregistrare în baza de date pentru noul client şi transmite înapoi o pagină HTML ce conţine numărul de înregistrare şi un număr de telefon de la serviciul de asistenţă. Aceasta nu este singura modalitate de a trata formularele, dar este o modalitate des întâlnită. Există un număr mare de cărţi despre scrierea scripturilor CGI şi programarea în Perl. Câteva exemple sunt: (Hanegan, 2001; Lash, 2002; şi Meltzer şi Michalski, 2001).

Scripturile CGI nu sunt singura modalitate de a genera conţinut dinamic la server. O altă modalitate des întâlnită este de a îngloba mici scripturi în paginile HTML şi a lăsa serverul să le execute pentru a genera pagina. Un limbaj popular pentru scrierea acestor scripturi este **PHP** (**PHP: Hypertext Procesor; rom.: Procesor Hipertext**). Pentru a fi folosit, serverul trebuie să înţeleagă PHP (exact cum programul de navigare trebuie să înţeleagă XML pentru a interpreta paginile de Web scrise în XML). De obicei, serverele se aşteaptă ca paginile de Web ce conţin PHP să aibă extensia *php* mai degrabă decât *html* sau *htm.*

Un mic script PHP este ilustrat în fig. 7-34; ar trebui să funcţioneze pe orice server care are PHP instalat. Conţine HTML normal cu excepţia scriptului PHP dintre marcajele <?php ... ?>. Ceea ce generează este o pagină de Web ce afişează ceea ce ştie despre programul de navigare care o apelează. Programele de navigare trimit de obicei o serie de informaţii odată cu cererea lor (şi orice cookie aplicabil) şi această informaţie este pusă în variabila *HTTP_USER_AGENT.* Când acest program este pus în fişierul *test.php* în directorul de WWW al companiei ABCD, tastând URL-ul *www.abcd.com/test.php* se va afişa o pagină de Web care spune utilizatorului ce program de navigare, ce limbă şi ce sistem de operare foloseşte.

```
<html> <body> <h2> This is what I know about you </h2> <?php echo
$HTTP_USER_AGENT ?>
</body> </html>
```

Fig. 7-34. O pagină HTML cu PHP înglobat

PHP este folositor în special la tratarea formularelor şi este mai simplu de utilizat decât scripturile CGI. Ca un exemplu al modului său de funcţionare, să considerăm exemplul din fig. 7-35(a). Această figură conţine o pagină HTML normală cu un formular în interior. Singurul lucru neobişnuit la această pagină este prima linie, care spune că fişierul *action.php* va fi invocat pentru a trata parametrii după ce utilizatorul a completat şi transmis formularul. Pagina afişează două căsuţe de text, una cerând numele şi cealaltă vârsta. După ce aceste căsuţe au fost completate şi formularul transmis, serverul analizează şirul de caractere de forma celui din fig. 7-30, punând numele în variabila *name* şi vârsta în variabila *age*. Începe apoi să proceseze fişierul *action.php*, arătat în fig. 7-35(b) pentru obţinerea răspunsului. În timpul procesării acestui fişier sunt executate comenzile PHP. Dacă utilizatorul a completat valorile „Barbara” şi „24” în căsuţele formularului, fişierul transmis înapoi este cel dat în fig. 7-35(c). Astfel, tratarea formularelor devine extrem de simplă în PHP.

Deşi PHP este uşor de utilizat, este de fapt un limbaj de programare puternic, orientat pe interfa- ţarea dintre Web şi o bază de date a serverului. Suportă variabile, şiruri de caractere, vectori şi majoritatea structurilor de control din C, dar un sistem de I/E mult mai puternic decât *printf*. PHP este un program public (open source) şi disponibil gratuit. A fost conceput special să lucreze bine cu Apache, care este de asemenea gratuit şi care este cel mai larg utilizat server de Web din lume. Pentru mai multe informaţii despre PHP, vezi (Valade, 2002).

Am văzut până acum două moduri diferite de a genera pagini HTML dinamice: script-urile CGI şi PHP-ul înglobat. Există şi o a treia tehnică, numită **JSP (JavaServer Pages)**, care este similară cu PHP, cu excepţia faptului că partea dinamică este scrisă în limbajul de programare Java în loc de PHP. Paginile ce folosesc această tehnică au în numele fişierului extensia *jsp.* O a patra tehnică, **ASP (Active Server Pages)**, este versiunea de la Microsoft a PHP şi JavaServer Pages. Pentru generarea conţinutului dinamic foloseşte limbajul de script proprietar al Microsoft-ului, Visual Basic Script. Paginile ce folosesc această tehnică au extensia *asp.* Alegerea dintre *PHP, JSP,* şi *ASP* are în general mai mult de-a face cu politici (open-souce vs. Sun vs. Microsoft) decât cu tehnologia, cele trei limbaje fiind comparabile. Colecţia de tehnologii pentru generarea din zbor a conţinutului este uneori denumită **HTML dinamic**.

Generare dinamică de pagini de Web la client

Scripturile CGI, PHP, JSP şi ASP rezolvă problema formularelor şi a interacţiunilor cu bazele de date din server. Toate pot să accepte informaţii care vin din formulare, să caute informaţii într-una sau mai multe baze de date şi să genereze pagini HTML cu rezultate. Ceea ce nu poate face nici unul dintre scripturi este să răspundă la mişcările mouse-ului sau să interacţioneze direct cu utilizatorii. În acest scop, este necesar ca scripturile să fie înglobate în paginile HTML care sunt executate mai degrabă pe maşina clientului, decât pe maşina serverului. Începând cu HTML 4.0, astfel de scripturi erau permise folosind marcajul < script >. Cel mai popular limbaj de script la client este **JavaScript**, aşa că o să aruncăm o scurtă privire asupra lui.

```
<html>
<body>
<form action="action.php" method="post">
<p> Introduceti numele: <input type="text" name="name"> </p> <p> Introduceti varsta:
<input type="text" name="age"> </p> <input type="submit">
</form>
</body>
</html>
```

(a)

```
<html>
<body>
<h1> Raspuns: </h1>
Hello <?php echo $name; ?>.
Prezicere: anul urmator veti avea <?php echo $age + 1; ?> ani.
</body>
</html>
```

(b)

```
<html>
<body>
<h1> Raspuns: </h1>
Buna, Barbara.
Prezicere: anul urmator veti avea 25 ani.
</body>
</html>
```

(c)

Fig. 7-35. (a) O pagină Web ce conţine un formular.
(b) Un script PHP pentru afişarea dinamică a form-ului.
(c) Ieşirea scriptului PHP când datele de intrare sunt „Barbara” şi respectiv 24.

```
<html>
<head>
<script language="javascript" type="text/javascript"> function response(test_form) { var
person = test_form.name.value;
var years = eval(test_form.age.value) + 1; document.open();
document.writeln(“<html> <body>"); document.writeln(“Hello “ + person + “.<br>");
document.writeln(“Prezicere: la anul vei avea “ + years + “."); document.writeln(“</body>
</html>");
document.close();
}
</script> </head>

<body> <form> Introduceţi numele: <input type="text" name="name"> <p>
Introduceti varsta: <input type="text" name="age">
<p>
<input type="button" value="submit" onclick="response(this.form)">
</form>
</body> </html>
```

Fig. 7-36. Folosirea JavaScript Pentru procesarea unui formular.

JavaScript este un limbaj de script, *foarte* inspirat din câteva idei ale limbajului de programare Java. Nu este cu siguranţă Java. Ca alte limbaje de script, el este un limbaj de nivel foarte înalt. De exemplu, într-o singură linie din JavaScript este posibil să se creeze o fereastră de dialog, să se aştepte introducerea de text şi să se memoreze şirul rezultat într-o variabilă. Astfel de caracteristici de nivel înalt fac din JavaScript un limbaj ideal pentru crearea de pagini Web interactive. Pe de altă parte, faptul că nu este standardizat şi că se modifică mai repede ca o muscă prinsă într-o maşină cu raze X, fac extrem de dificilă scrierea de programe JavaScript care să funcţioneze pe toate platformele, dar poate într-o zi se va stabiliza.

Un exemplu de program în JavaScript, este cel din fig. 7-36. Ca şi în fig. 7-35(a), apare un formular în care se cer numele şi vârsta şi care calculează vârsta persoanei în anul următor. Corpul este aproape la fel ca în exemplul PHP, principala diferenţă fiind declararea butonului de trimitere a datelor şi asocierea unei funcţii cu acest buton. Această funcţie spune programului de navigare să invoce scriptul *response* la o apăsare de buton şi să-i trimită formularul ca

parametru.

Complet nouă aici este declararea funcţiei JavaScript *response* în antetul fişierului HTML, o zonă în mod normal rezervată titlurilor, culorilor de fundal şi aşa mai departe. Această funcţie extrage valoarea câmpului *name* din formular şi o păstrează ca şir în variabila *person.* De asemenea extrage valoarea câmpului *age,* o converteşte la un întreg prin folosirea funcţiei *eval,* o incrementează cu 1 şi reţine rezultatul în *years.* Apoi deschide un document pentru ieşire în care face trei scrieri, folosind metoda *writeln,* şi închide documentul. Documentul este un fişier HTML, după cum se poate vedea din diversele marcaje HTML din el. Programul de navigare afişează apoi documentul pe ecran.

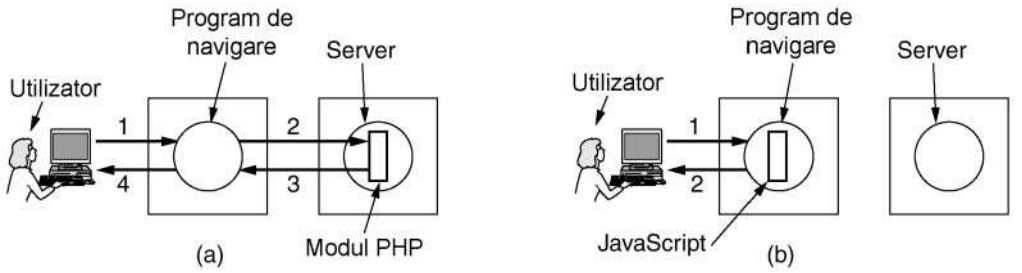


Fig. 7-37. (a) Scripting la server cu PHP. (b) Scripting la client cu JavaScript.

Este foarte important de înţeles că în timp ce fig. 7-35 şi fig. 7-36 arată similar, ele sunt procesate total diferit. În fig. 7-35, după ce utilizatorul a apăsat butonul *submit,* programul de navigare strânge informaţia într-un şir lung, în stilul celui din fig. 7-30 şi îl trimite serverului care a trimis pagina. Serverul vede numele fişierului PHP şi îl execută. Scriptul PHP produce o nouă pagină HTML şi acea pagină este trimisă înapoi programului de navigare pentru afişare. Cu fig. 7-36, când butonul *submit* este apăsat, programul de navigare interpretează o funcţie JavaScript conţinută pe pagină. Totul este făcut local, în programul de navigare. Nu se face nici un contact cu serverul. Ca o consecinţă, rezultatul este tipărit teoretic instantaneu, în timp ce cu PHP, poate exista o întârziere de câteva secunde înainte ca HTML-ul rezultat să ajungă la client. Diferenţa între utilizarea scripturilor la server şi utilizarea acestora la client este ilustrată în fig. 7-37, inclusiv paşii implicaţi. În ambele cazuri, paşii numerotaţi încep după afişarea formularului. Pasul 1 constă din acceptarea datelor de intrare ale utilizatorului. Apoi urmează procesarea acestora, care diferă în cele două cazuri.

```
<html>
<head>
<script language="javascript" type="text/javascript">
function response(test_form) { function factorial(n) { if (n==0) return 1; else return n *
factorial(n-l); } var r = eval(test_form.number.value);       // r = argument introdus de la
tastatura
document.myform.mytext.value = "Aici sunt rezultatele.\n";
for (var i = 1; i <= r; i++)                                  // tipareste o linie de la 1 la r
document.myform.mytext.value += (i + "! = " + factorial(i) + „\n");
}
```

```
</script> </head>
<body>
<form name="myform">
Introduceti un numar: <input type="text" name="number"> <input type="button"
value="calcul factorial" onclick="response(this.form)"> <p> <textarea name="mytext"
rows=25 cols=50> </textarea>
</form> </body> </html>
```

Fig. 7-38. Un program JavaScript pentru calculul şi afişarea factorialului.

Această diferenţă nu înseamnă că JavaScript este mai bun ca PHP. Utilizările lor sunt complet diferite. PHP (şi, prin implicaţie, JSP şi ASP) sunt utilizate când este necesară o interacţiune cu o bază de date aflată la distanţă. JavaScript este utilizat când interacţiunea se face cu utilizatorul la calculatorul clientului. Este cu siguranţă posibil (şi des întâlnit) să existe pagini HTML care folosesc atât PHP cât şi JavaScript, deşi evident nu pot face acelaşi lucru pe acelaşi buton, sau să deţină acelaşi buton.

JavaScript este un limbaj de programare matur, cu toată puterea limbajelor C şi Java. Are variabile, şiruri, vectori, obiecte, funcţii, şi toate structurile de control obişnuite. Are, de asemenea, un număr mare de facilităţi specifice paginilor Web, inclusiv abilitatea de a lucra cu ferestre şi cadre, setarea şi obţinerea de cookie-uri, lucrul cu formulare, şi cu hiper-legături. Un exemplu de program JavaScript care utilizează o funcţie recursivă este dat în fig. 7-38.

JavaScript poate, de asemenea, să urmărească mişcarea mouse-ului peste obiectele afişate. Multe pagini Web ce conţin JavaScript au proprietatea că atunci când mouse-ul se mişcă peste un text sau o imagine, se întâmplă ceva. Deseori, imaginea se schimbă sau apare dintr-o dată un meniu. Acest tip de comportament este uşor de programat în JavaScript şi conduce la pagini de Web foarte dinamice. În fig. 7-39 este dat un exemplu.

```
<html>
<head>
<script language="javascript" type="text/javascript"> if (!document.myurl) document.myurl =
new Array();
document.myurl[0] = “http://www.cs.vu.nl/ast/im/kitten.jpg"; document.myurl[1] =
“http://www.cs.vu.nl/ast/im/puppy.jpg"; document.myurl[2] =
“http://www.cs.vu.nl/ast/im/bunny.jpg";
function pop(m) {
        var urx = “http://www.cs.vu.nl/ast/im/cat.jpg";
        popupwin = window.open(document.myurl[m], “mywind", “width=250,height=250");
}
</script> </head>
<body>
<p> <a href="#" onMouseover="pop(0); return false;"> Kitten </a> </p> <p> <a href="#"
onMouseover="pop(1); return false;"> Puppy </a> </p> <p> <a href="#"
onMouseover="pop(2); return false;"> Bunny </a> </p> </body> </html>
```

Fig. 7-39. O pagină Web interactivă care răspunde la mişcarea mouse-ului.

JavaScript nu este singura cale de a face paginile Web foarte interactive. Altă

metodă populară este bazată pe folosirea **applet**-urilor. Acestea sunt mici programe Java care au fost compilate într- un cod maşină pentru un calculator virtual numit **JVM (Java Virtual Machine –** Maşina Virtuală Java). Applet-urile pot fi incluse în paginile HTML (între <applet> şi </applet>) şi sunt interpretate de programe de navigare care cunosc JVM. Deoarece applet-urile nu sunt executate, ci interpretate, interpretorul Java poate să le împiedice să facă Lucruri Rele. Cel puţin în teorie. În practică, autorii de applet-uri au găsit şi exploatat un şir aproape nesfârşit de erori în bibliotecile Java de I/E.

Răspunsul Microsoft la applet-urile Java de la Sun au fost paginile Web cu **controale Active-X (Active-X controls)**, care sunt programe compilate pentru o maşină Pentium şi sunt executate direct în hardware. Această proprietate le face mult mai rapide şi mai flexibile decât applet-urile interpretate, pentru că pot face orice poate face un program. Când Internet Explorer vede un control Acti- ve-X într-o pagină Web, îl descarcă, îi verifică identitatea şi îl execută. Totuşi, descărcarea şi executarea de programe străine ridică probleme de securitate, la care ne vom referi în cap. 8.

Din moment ce aproape toate programele de navigare pot să interpreteze atât programe Java cât şi JavaScript, un programator care vrea să facă o pagină Web foarte interactivă va putea să aleagă între două tehnici, iar dacă nu se doreşte portabilitatea pe mai multe platforme, poate să aleagă şi Active-X. Ca o regulă generală, programele JavaScript sunt mai uşor de scris, applet-urile Java se execută mai rapid iar controalele Active-X cel mai rapid dintre toate. De asemenea, din moment ce toate programele de navigare implementează exact aceeaşi JVM, dar nu există două programe de navigare care să ştie aceeaşi versiune de JavaScript, applet-urile Java sunt mai portabile decât programele JavaScript. Pentru mai multe detalii despre JavaScript, există multe cărţi, fiecare cu multe (deseori peste 1000) pagini. Câteva exemple sunt (Easttom, 2001; Harris 2001; şi McFerdries, 2001).

Înainte de a părăsi subiectul conţinutului dinamic al Web-ului, să recapitulăm pe scurt ce am atins până acum. Pagini Web complete se pot genera din mers, folosind diverse script-uri pe maşina server. Odată ce sunt primite de programul de navigare, ele sunt tratate ca pagini HTML normale şi sunt doar afişate. Script-urile pot fi scrise în Perl, PHP, JSP sau ASP, după cum este arătat în fig. 7-40.

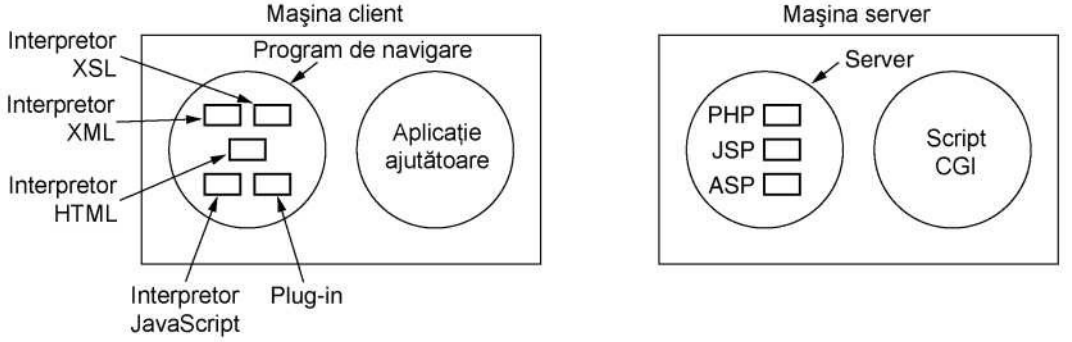


Fig. 7-40. Diverse moduri de a genera şi afişa conţinut.

Generarea conţinutului dinamic este posibilă şi în partea clientului. Paginile Web pot fi scrise în XML şi convertite la HTML conform unui fişier XSL. Programele JavaScript pot să efectueze diverse calcule. În sfârşit, plug-în-urile (plug-ins) şi aplicaţiile ajutătoare (helper applications) pot fi folosite pentru afişarea conţinutului într-o varietate de forme.

7.3.4 HTTP – HyperText Transfer Protocol

Protocolul de transfer utilizat pe Web este **HTTP (HyperText Transfer Protocol, rom.:Protocol de Transfer al Hipertextului)**. Acesta specifică ce mesaje pot trimite clienţii către servere şi ce răspunsuri primesc înapoi. Fiecare interacţiune constă dintr-o cerere ASCII, urmată de un răspuns MIME conform RFC 822. Toţi clienţii şi toate serverele trebuie să se supună acestui protocol. Este definit în RFC 2616. În această secţiune vom trata câteva din proprietăţile sale cele mai importante.

Conexiuni

Modul uzual prin care un program de navigare contactează un server este de a stabili o conexiune TCP pe portul 80 pe maşina serverului, deşi această procedură nu este cerută formal. Avantajul de a folosi TCP este că nici programele de navigare şi nici serverele nu trebuie sa se preocupe de mesajele pierdute, mesajele duplicate, mesajele lungi, sau mesajele de confirmare. Toate aceste aspecte sunt tratate de implementarea TCP.

În HTTP 1.0, după ce conexiunea era stabilită, o singură cerere era transmisă şi un singur răspuns era primit înapoi. Apoi conexiunea TCP era eliberată. Într-o lume în care pagina tipică Web consta în întregime din text HTML, această metodă era adecvată. În câţiva ani însă, o pagină medie Web conţinea un număr mare de iconiţe, imagini şi alte lucruri plăcute vederii, astfel că stabilirea unei conexiuni TCP pentru a prelua o singură iconiţă a devenit un mod foarte costisitor de a opera.

Această observaţie a dus la apariţia Hl'l'P 1.1, care suportă **conexiuni persistente**. Cu ele, este posibilă stabilirea unei conexiuni TCP, trimiterea unei cereri şi obţinerea unui răspuns, apoi trimiterea unor cereri adiţionale şi obţinerea de răspunsuri adiţionale. Prin distribuirea pornirii şi eliberării unei conexiuni TCP peste mai multe cereri, supraîncărcarea relativă datorată TCP-ului este mult mai mică pe fiecare cerere. Este de asemenea posibilă trimiterea cererilor prin mecanismul pipeline, adică trimiterea cererii 2 înainte ca răspunsul la cererea 1 să fi sosit.

Metode

Cu toate că HTTP a fost proiectat pentru utilizarea în Web, el a fost creat intenţionat mai general decât era necesar în perspectiva aplicaţiilor orientate pe obiecte. Pentru aceasta sunt suportate operaţiile, denumite **metode**, care fac mai mult decât cele care doar cer o pagină Web. Această consideraţie generală a permis apariţia SOAP. Fiecare cerere constă din una sau mai multe linii de text ASCII, în care primul cuvânt din prima linie este numele metodei cerute. Metodele incorporate sunt listate în fig. 7-41. Pentru accesarea unor obiecte generale, metode adiţionale specifice obiectelor pot fi de asemenea disponibile. În numele

metodelor este semnificativă utilizarea literelor mari/mici, de exemplu *GET* este o metodă acceptată, dar nu şi *get.*

Metoda	Descriere
GET	Cerere de citire a unei pagini Web
HEAD	Cerere de citire a antetului unei pagini de Web
PUT	Cerere de memorare a unei pagini de Web
POST	Adăugarea la o resursă anume (de exemplu o pagină
DELETE	Ştergerea unei pagini de Web
TRACE	Tipărirea cererii care a sosit
CONNECT	Rezervat pentru o utilizare în viitor
OPTIONS	Interogarea anumitor opţiuni

Fig. 7-41. Metode de cerere standard pentru HTTP.

Metoda *GET* cere serverului să trimită pagina (prin care noi înţelegem obiect, în cel mai general caz, dar în practică de obicei doar un fişier). Pagina este codată corespunzător în MIME. Marea majoritate a cererilor către servere Web sunt metode *GET*. Forma uzuală a metodei *GET* este

```
GET fişier HTTP-1.1
```

unde *fişier* denumeşte resursa (fişierul) ce va fi adusă, şi 1.1 este versiunea de protocol utilizat.

Metoda *HEAD* cere doar antetul mesajului, fără să ceară şi pagina propriu-zisă. Această metodă poate să fie utilizată pentru a afla când s-a făcut ultima modificare, pentru a obţine informaţii pentru indexare, sau numai pentru a verifica corectitudinea unui URL.

Metoda *PUT* este inversa metodei *GET*: în loc să citească o pagină, o scrie. Această metodă permite crearea unei colecţii de pagini de Web pe un server la distanţă. Corpul cererii conţine pagina. Pagina poate să fie codificată utilizând MIME, caz în care liniile care urmează după *PUT* pot include *Content-Type* şi antete de autentificare, pentru a demonstra că într-adevăr cel care face cererea are dreptul de a realiza operaţia cerută.

Similară metodei *PUT* este metoda *POST*. Şi ea conţine un URL, dar în loc să înlocuiască date existente, noile date se vor adăuga într-un mod generalizat. De exemplu, se poate transmite un mesaj la un grup de ştiri sau adăuga un fişier la un sistem de informare în reţea. În practică, nici *PUT* şi nici *POST* nu sunt utilizate prea mult.

DELETE realizează ce era de aşteptat: ştergerea unei pagini. Ca şi la *PUT,* autentificarea şi drepturile de acces joacă aici un rol important. Nu există nici o garanţie asupra succesului operaţiei *DELETE*, deoarece chiar dacă serverul doreşte să execute ştergerea, fişierul poate să aibă atribute care să interzică serverului HTTP să îl modifice sau să îl şteargă.

Metoda *TRACE* este pentru verificarea corectitudinii. Ea cere serverului să trimită înapoi cererea. Această metodă este utilă când cererile nu sunt procesate corect şi clientul vrea să ştie ce fel de cerere a ajuns de fapt la server.

Metoda *CONNECT* nu este utilizată în prezent. Este rezervată pentru utilizări ulterioare.

Metoda *OPTIONS* asigură o modalitate pentru client de a interoga serverul despre proprietăţile acestuia sau despre cele ale unui anumit fişier.

Fiecare cerere obţine un răspuns ce constă din linia de stare şi posibile informaţii suplimentare (de exemplu, o parte sau toată pagina Web). Linia de stare conţine un cod de stare de trei cifre, indicând dacă cererea a fost satisfăcută şi dacă nu, cauza. Prima cifră este utilizată pentru împărţirea răspunsurilor în cinci mari grupuri, ca în fig. 7-42. Codurile 1xx sunt utilizate în practică foarte rar. Codurile 2xx indică tratarea cu succes a cererii şi conţinutul (dacă există) este returnat. Codurile 3xx spun clientului să caute în altă parte, prin folosirea unui URL diferit, sau în propria memorie ascunsă (discutată mai târziu). Codurile 4xx indică insuccesul cererii din cauza unei erori la client, precum o cerere invalidă sau o pagină inexistentă. În fine, erorile 5xx indică o problemă în server, datorată codului său sau unei supraîncărcări temporare.

Cod	Semnificaţie	Exemple
1xx	Informaţie	100 = serverul acceptă tratarea cererii de la client
2xx	Succes	200 = cerere reuşită; 204 = nu există conţinut
3xx	Redirectare	301 = pagină mutată; 304 = pagina din memoria ascunsă este încă validă
4xx	Eroare la client	403 = pagină interzisă; 404 = pagina nu a fost găsită
5xx	Eroare la server	500 = eroare internă la server; 503 = încearcă mai

Fig. 7-42. Grupuri de răspunsuri ale codurilor de stare.

Antete de mesaje

Linia de cerere (de exemplu linia cu metoda *GET*) poate fi urmată de linii adiţionale cu mai multe informaţii. Acestea poartă numele de **antete de cerere**. Această informaţie poate fi comparată cu parametrii unui apel de procedură. Răspunsurile pot avea de asemenea **antete de răspuns**. Anumite antete pot fi folosite în orice sens. O selecţie a celor mai importante este dată în fig. 7-43.

Antetul *User-Agent* permite clientului să informeze serverul asupra programului său de navigare, sistemului de operare şi altor proprietăţi. În fig. 7-34 am văzut că serverul avea în mod magic această informaţie şi că o poate obţine la cerere într-un script PHP. Antetul este utilizat de client pentru a-i asigura serverului această informaţie.

Antet	Tip	Descriere
User-Agent	Cerere	Informaţie asupra programului de navigare şi a
Accept	Cerere	Tipul de pagini pe care clientul le poate trata
Accept-Charset	Cerere	Seturile de caractere care sunt acceptabile la client
Accept-Encoding	Cerere	Codificările de pagini pe care clientul le poate trata
Accept-Language	Cerere	Limbajele naturale pe care clientul le poate trata
Host	Cerere	Numele DNS al serverului
Authorization	Cerere	O listă a drepturilor clientului
Cookie	Cerere	Trimite un cookie setat anterior înapoi la server
Date	Ambele	Data şi ora la care mesajul a fost trimis
Upgrade	Ambele	Protocolul la care transmiţătorul vrea să comute
Server	Răspuns	Informaţie despre server
Content-Encoding	Răspuns	Cum este codat conţinutului (de exemplu, gzip)
Content-Language	Răspuns	Limbajul natural utilizat în pagină
Content-Length	Răspuns	Lungimea paginii în octeţi

Content-Tzpe	Răspuns	Tipul MIME al paginii
Last-Modified	Răspuns	Ora şi data la care pagina a fost ultima dată
Location	Răspuns	O comandă pentru client pentru a trimite cererea în
Accept-Ranges	Răspuns	Serverul va accepta cereri în anumite limite de octeţi
Set-Cookie	Răspuns	Serverul vrea să salveze un cookie la client

Fig. 7-43. Câteva antete de mesaje HTTP.

Cele patru antete *Accept* spun serverului ce este dispus clientul să accepte în cazul în care acesta are un repertoriu limitat despre ceea ce este acceptabil. Primul antet specifică ce tipuri MIME sunt acceptate (de exemplu, text/html). Al doilea reprezintă setul de caractere (de exemplu ISO-8859 sau Unicode-1-1). Al treilea se referă la metode de compresie (de exemplu, gzip). Al patrulea indică un limbaj natural (de exemplu, spaniola). Dacă serverul are mai multe pagini din care poate să aleagă, el poate utiliza această informaţie pentru a furniza clientului pagina pe care o caută. Dacă nu poate satisface cererea, este întors un cod de eroare şi cererea eşuează.

Antetul *Host* denumeşte serverul. El este luat din URL. Antetul este obligatoriu. Este utilizat deoarece anumite adrese IP pot servi mai multe nume de DNS şi serverul are nevoie de o anumită modalitate de a spune cărui calculator să-i trimită cererea.

Antetul *Authorization* este necesar pentru protecţia paginilor. În acest caz, clientul trebuie să demonstreze că are dreptul de a vedea pagina cerută. Acest header este utilizat în acest scop.

Deşi cookie-urile sunt tratate în RFC 2109 mai mult decât în RFC 2616, şi ele au două antete. Antetul *Cookie* este utilizat de clienţi pentru a întoarce serverului un cookie care a fost anterior trimis de o maşină aflată în domeniul serverului.

Antetul *Date* poate fi utilizat în ambele sensuri şi conţine ora şi data la care a fost trimis mesajul. Antetul *Upgrade* este folosit pentru a face mai uşoară crearea unei tranziţii către o viitoare (posibil incompatibilă) versiune a protocolului HTTP. Acesta permite clientului, să anunţe ce anume suportă, şi serverului să afirme ceea ce foloseşte.

Acum am ajuns la antetele utilizate exclusiv de către server în răspunsul cererilor. Primul, *Server,* permite serverului să spună cine este şi câteva proprietăţi, dacă doreşte.

Următoarele patru antete, toate începând cu *Content-,* permit serverului să descrie proprietăţile paginii pe care o transmite.

Antetul *Last-Modified* spune când a fost modificată ultima dată pagina. Acest antet joacă un rol important în mecanismul de memorie ascunsă.

Antetul *Location* este utilizat de server pentru a informa clientul că ar trebui să utilizeze un alt URL. Acesta poate fi folosit dacă pagina a fost mutată, sau pentru a da permisiunea mai multor URL-uri de a referi aceeaşi pagină (posibil pe servere diferite). Este de asemenea utilizată pentru companiile care au o pagină de Web principală în domeniul *com*, dar care redirecţionează clienţii la o pagină naţională sau regională în funcţie de adresa lor IP sau limba preferată.

Dacă o pagină este foarte mare, un client mic poate nu o doreşte dintr-o dată.

Unele servere acceptă cereri în anumite intervale de octeţi, astfel că pagina poate fi citită în mai multe unităţi mai mici. Antetul *Accept-Ranges* anunţă asentimentul severului de a trata acest tip de cerere parţială de pagini.

Al doilea antet pentru cookie, *Set-Cookie*, se referă la modul în care serverele trimit cookie-uri la clienţi. Este de aşteptat salvarea cookie-ului de către client şi returnarea acestuia la cereri ulterioare ale serverului.

Exemplu de utilizare HTTP

Deoarece HTTP este un protocol ASCII, este destul de uşor pentru o persoana aflată la un terminal (ca opus al programului de navigare) să vorbească direct cu serverele de Web. Este necesară doar o conexiune TCP la portul 80 pe server. Cititorii sunt încurajaţi să încerce personal acest scenariu (preferabil dintr-un sistem UNIX, deoarece anumite sisteme nu returnează starea conexiunii).

```
Trying 4.17.168.6...
Connected to www.ietf.org.
Escape character is '^A]'.
HTTP/1.1 200 OK
Date: Wed, 08 May 2002 22:54:22 GMT
Server: Apache/1.3.20 (Unix) mod_ssl/2.8.4 OpenSSL/0.9.5a
Last-Modified: Mon, 11 Sep 2000 13:56:29 GMT
ETag: "2a79d-c8b-39bce48d"
Accept-Ranges: bytes Content-Length: 3211 Content-Type: text/html X-Pad: avoid browser bug
<html>
<head>
<title>IETF RFC Page</title>
<script language="javascript"> function url() {
var x = document.form1.number.value
if (x.length == 1) { x = "000" + x } if (x.length == 2) { x = "00" + x } if (x.length == 3) { x =
"0" + x } document.form1.action = „/rfc/rfc" + x + „.txt" document.form1.submit
}
</script>
</head>
```

Fig. 7-44. Primele linii ale fişierului *www.ietf.org/rfc.html*.

Următoarea secvenţă de comenzi va realiza acest lucru:

```
telnet www.ietf.org 80 >log
GET /rfc.html HTTP/1.1
Host: www.ietf.org
close
```

Această secvenţă de comenzi porneşte o conexiune telnet (adică TCP) pe portul 80 al serverului Web al IETF, *www.ietf.org*. Rezultatul sesiunii este redirectat către fişierul *log* pentru o inspecţie ulterioară. Apoi urmează comanda *GET* denumind fişierul şi protocolul. Următoarea linie este antetul obligatoriu *Host.* Linia rămasă liberă este de asemenea cerută. Ea semnalează serverului că nu mai sunt antete de cerere. Comanda *close* indică programului de telnet să întrerupă conexiunea.

Fişierul *log* poate fi inspectat folosind un editor. Acesta ar trebui să înceapă similar cu liniile de cod din fig. 7-44, cu excepţia unei modificări recente de către

IETF.

Primele trei linii reprezintă ieşirea programului telnet, şi nu de la serverul la distanţă. Linia ce începe cu HTTP/1.1 este răspunsul IETF prin care spune că este dispus să vorbească HTTP/1.1 cu tine. Apoi urmează un număr de antete şi apoi conţinutul. Am văzut deja toate antetele, cu excepţia lui *ETag* care este un identificator de pagină unic, referitor la memoria ascunsă, şi *X-Pad* care nu este standardizat, probabil o cale de scăpare pentru vreun program de navigare cu erori.

7.3.5 Îmbunătăţiri ale performanţei

Popularitatea Web-ului aproape că a fost propria sa distrugere. Servere, rutere şi linii folosite de Web sunt adesea supraîncărcate. Multă lume a început să denumească WWW-ul ca World Wide Wait (rom: aşteptare de întindere planetară). Ca o consecinţă a acestor întârzieri fără sfârşit, cercetătorii au dezvoltat diverse tehnici pentru îmbunătăţirea performanţelor. Vom examina acum trei dintre ele: memoria ascunsă, replicarea serverelor şi reţele de livrare a conţinutului.

Memoria ascunsă

Un mod simplu de a îmbunătăţi performanţa este de a salva paginile care au fost cerute pentru cazul în care ele vor fi utilizate din nou. Această tehnică este efectivă în special pentru paginile care sunt vizitate foarte mult, ca *www.yahoo.com* şi *www.cnn.com*. Paginile pot fi păstrate pentru utilizări ulterioare în **memoria ascunsă (eng.: cache)**. Procedura uzuală este ca un proces, denumit **proxy (rom.: delegat)**, să întreţină această memorie. Pentru a utiliza memoria ascunsă, un program de navigare poate fi configurat să adreseze toate cererile de pagini proxy-ului, şi nu serverului real unde se află pagina respectivă. Dacă proxy-ul are pagina, o returnează imediat. Dacă nu, aduce pagina de la server, o adaugă în memoria ascunsă pentru utilizarea ulterioară şi o întoarce clientului care a cerut-o.

Două întrebări importante aferente memoriei ascunse sunt:

1. Cine ar trebui să deţină memoria ascunsă?
2. Cât de mult timp ar trebui să stea paginile în memoria ascunsă?

Există mai multe răspunsuri la prima întrebare. PC-urile individuale de obicei rulează proxy-uri, deci pot să caute rapid pagini vizitate anterior. Într-un LAN al unei companii, proxy-ul este în general o maşină ce poate fi accesată de toate maşinile din acel LAN, astfel că dacă un utilizator se uită la o anumită pagină şi apoi alt utilizator din acelaşi LAN vrea aceeaşi pagină, ea poate fi adusă din memoria ascunsă a proxy-ului. Multe ISP-uri rulează proxy-uri, pentru a accelera accesul clienţilor

lor. De obicei toate memoriile ascunse operează în acelaşi timp, deci cererile se duc iniţial la proxy- ul local. Dacă eşuează, proxy-ul local cere pagina proxy-ului din LAN. Dacă şi aceasta eşuează, proxy-ul din LAN încearcă la proxy-ul ISP-ului. Ultimul trebuie să reuşească să aducă paginile, fie din memoria sa ascunsă, fie de la o memorie ascunsă de nivel superior, fie de la însuşi serverul ce deţine pagina. O schemă ce include mai multe memorii ascunse care pot fi încercate în secvenţă este

denumită **memorie ascunsă ierarhică (hierarchical caching)**. O posibilă implementare este ilustrată în fig. 7-45.

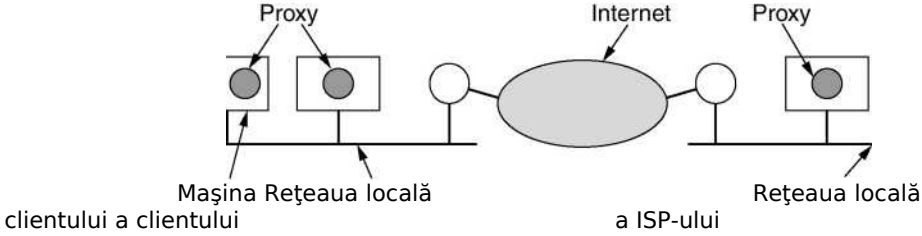


Fig. 7-45. Memorie ascunsă ierarhică cu 3 proxy-uri .

Cât timp ar trebui paginile să rămână în memoria ascunsă este un pic mai dificil de aflat. Anumite pagini nu ar trebui să fie păstrate deloc în memoria ascunsă. De exemplu, o pagină ce conţine preţurile pentru cele mai active 50 de acţiuni la bursă se schimbă la fiecare secundă. Dacă s-ar păstra în memoria ascunsă, un utilizator ce ia o astfel de copie ar lua date **vechi** (adică depăşite). Pe de altă parte, din momentul în care schimbul de acţiuni s-a închis pe ziua respectivă, pagina va rămâne validă ore sau zile, până când începe următoarea licitaţie. Astfel, eficienţa menţinerii unei pagini în memoria ascunsă poate varia foarte mult în timp.

Problema-cheie în determinarea eliminării unei pagini din memoria ascunsă este legată de vechimea pe care utilizatorii sunt dispuşi să o accepte (din moment ce paginile din memoria ascunsă sunt ţinute pe disc, cantitatea de memorare consumată reprezintă rareori o problemă). Dacă un proxy elimină repede paginile, el va întoarce rar o pagină veche, dar nu va fi prea eficient (adică va avea o rată scăzută de succes). Dacă păstrează paginile prea mult, poate avea o rată mai mare dar cu preţul de a întoarce deseori pagini cu informaţie expirată.

Există două abordări în tratarea acestei probleme. Prima utilizează o euristică pentru a şti cât timp să menţină fiecare pagină. O euristică des întâlnită este cea în care se ţine cont de antetul *Last- Modified* (vezi fig. 7-43). Dacă o pagină a fost modificată cu o oră în urmă, se ţine în memoria ascunsă o oră. Dacă a fost modificată cu un an în urmă, este evident o pagină foarte veche (de exemplu, o listă a zeilor din mitologia greacă şi cea romană), deci poate fi păstrată în memoria ascunsă pentru un an, cu o probabilitate rezonabilă că nu se va modifica în decursul anului. Deşi această euristică funcţionează bine în practică, ea întoarce, totuşi, pagini vechi din când în când.

Cealaltă abordare este mai costisitoare dar elimină posibilitatea paginilor vechi prin utilizarea unor caracteristici speciale ale RFC 2616 care tratează administrarea memoriei ascunse. Una din cele mai utilizate caracteristici este antetul de cerere *If-Modified-Since,* pe care un proxy poate să-l trimită unui server. El specifică pagina pe care o vrea proxy-ul şi momentul la care pagina din memoria ascunsă a fost modificată ultima dată (din antetul *Last-Modified*). Dacă pagina nu a mai fost modificată de atunci, serverul trimite înapoi un scurt mesaj *Not Modified* (codul de stare 304 din fig. 7-42), care indică proxy-ului să utilizeze pagina din memoria ascunsă. Dacă pagina a mai fost modificată de atunci, este returnată noua pagină. În timp ce pentru această abordare este întotdeauna nevoie de un mesaj de

cerere şi unul de răspuns, mesajul de răspuns va fi foarte scurt când intrarea în memoria ascunsă este încă validă.

Aceste două abordări pot fi combinate uşor. Pentru primul AT după aducerea paginii, proxy-ul doar returnează pagina clienţilor ce o cer. După ce pagina a stat un timp în memoria ascunsă, proxy- ul utilizează mesajul *If-Modified-Since* pentru a verifica valabilitatea acesteia. Alegerea AT implică invariabil o euristică, depinzând de cât de mult timp a trecut de la modificarea paginii.

Paginile Web cu conţinut dinamic (de exemplu, cele generate de un script PHP) nu ar trebui niciodată păstrate în memoria ascunsă, deoarece parametrii pot fi diferiţi la următoarea accesare. Pentru a trata aceasta şi alte cauze, există un mecanism general prin care un server instruieşte toate proxy-urile până la client să nu folosească din nou pagina curentă până nu îi verifică valabilitatea. Acest mecanism poate fi folosit de asemenea pentru orice pagină pasibilă să fie modificată curând. Diverse mecanisme de control al memoriei ascunse sunt definite în RFC 2616.

Mai există o abordare în îmbunătăţirea performanţei, memoria ascunsă proactivă. Când un proxy aduce o pagină de la un server, el poate inspecta pagina pentru a vedea dacă ea conţine hiper- legături. Dacă este aşa, poate să lanseze cereri serverelor corespunzătoare pentru a preîncărca în memoria ascunsă paginile la care punctează, pentru cazul în care va fi nevoie de ele. Această tehnică poate reduce timpul de acces pentru cererile ulterioare, dar poate, la fel de bine, să inunde liniile de comunicaţie cu pagini care nu vor fi niciodată necesare.

În mod clar, memoria ascunsă în Web este departe de a fi banală. Mult mai multe se pot spune despre ea. De fapt, s-au scris cărţi întregi pe această temă, de exemplu (Rabinovich şi Spatscheck, 2002; şi Wessels, 2001). Dar este timpul ca noi să trecem la un alt subiect.

Replicarea serverelor

Memoria ascunsă este o tehnică orientată spre client pentru îmbunătăţirea performanţelor, dar există şi tehnici orientate pe server. Cea mai întâlnită abordare pentru îmbunătăţirea performanţelor serverelor este replicarea conţinutului lor în mai multe locuri separate. Această tehnică este câteodată denumită **oglindire (mirroring)**.

Într-o utilizare tipică a oglindirii într-o companie, pagina principală de Web conţine câteva imagini cu legături către siturile Web regionale, de exemplu siturile din est, vest, nord şi sud. Utilizatorul urmează legătura cea mai apropiată. Din acel moment, toate cererile se duc la serverul selectat.

Siturile oglindite sunt în general complet statice. Compania decide unde să plaseze copiile, dispune de un server în fiecare regiune, şi plasează (mai mult sau mai puţin) întregul conţinut în fiecare loc (omiţând, probabil, dezăpezitoarele în situl de la Miami şi şezlongurile în situl din Anchorage). Alegerea siturilor rămâne în general stabilă luni sau chiar ani de zile.

Din păcate, Web-ul prezintă un fenomen cunoscut ca **aglomerare bruscă (flash crowds)** în care un sit Web care era anterior necunoscut, nevizitat, aproape mort, devine dintr-o dată centrul universului. De exemplu, până pe 6 noiembrie 2000, situl Web al secretarului de stat din Florida, *www.dos.state.fl.us*, informa tacit

despre întâlnirile cabinetului statului Florida şi oferea instrucţiuni pentru a deveni notar în Florida. Dar pe 7 noiembrie 2000, când preşedinţia Statelor Unite depindea dintr-o dată de câteva mii de voturi disputate în câteva provincii din Florida, a devenit unul din primele 5 situri Web din lume. Evident, nu a putut suporta încărcarea şi aproape că a murit strivit sub ea.

Este necesar un mod prin care un sit Web, ce observă dintr-o dată o cerere masivă a traficului, să se cloneze automat în atâtea locaţii cât este necesar şi să păstreze aceste situri operaţionale până când trece furtuna, moment în care opreşte majoritatea sau chiar totalitatea acestora. Pentru a avea această abilitate, un sit are nevoie de o înţelegere prealabilă cu o companie care deţine mai multe situri, prin care se pot crea replici la cerere şi pentru care se plăteşte în funcţie de capacitatea pe care o foloseşte în realitate.

O strategie şi mai flexibilă este de a crea replici dinamice la nivel de pagini, în funcţie de unde vine traficul. Câteva cercetări pe această temă sunt raportate în (Pierre ş.a.., 2001; şi Pierre ş.a.., 2002).

Reţele de livrare de conţinut

Culmea capitalismului este că cineva a descoperit cum să câştige bani din World Wide Wait. Funcţionează în felul următor. Companiile denumite **CDN (Content Delivery Networks, rom: reţele de livrare de conţinut)** vorbesc cu deţinătorii conţinutului (situri muzicale, ziare, şi alţii care doresc să facă disponibil conţinutul uşor şi rapid) şi se oferă să livreze acest conţinut utilizatorilor finali în mod eficient, contra cost. După semnarea contractului, deţinătorul conţinutului oferă CDN-ului conţinutul sitului său Web pentru preprocesare (care va fi discutată imediat) şi apoi distribuţie.

Apoi CDN vorbeşte cu un număr mare de ISP-uri şi se oferă să îi plătească bine pentru dreptul de a plasa un server administrat la distanţă, plin de conţinut valoros, pe LAN-urile lor. Nu numai că este o sursă de venit, dar asigură de asemenea clienţilor ISP-urilor timpi de răspuns excelenţi pentru a ajunge la conţinutul CDN-ului, oferind astfel ISP-ului un avantaj competitiv faţă de alte ISP-uri care nu au acceptat oferta CDN-ului. În aceste condiţii, colaborarea cu un CDN este ceva la mintea cocoşului pentru ISP. Ca o consecinţă, cele mai mari CDN-uri au mai mult de 10.000 de servere răspândite în toată lumea.

```
<html>
<head><title>Furry Video</title></head>
<body>
<h1>Furry Video's Product List</h1>
<p>Click below for free samples.</p>
<a href="bears.mpg">Bears Today</a><br>,
<a href="bunnies.mpg">Funny Bunnies</a><br>
<a href="mice.mpg">Nice Mice</a><br>
</body>
</html>
```

(a)

```
<html>
<head><title>Furry Video</title></head>
```

```
<body>
<h1>Furry Video's Product List</h1>
<p>Click below for free samples.</p>
<a href="http://cdn-server.com/furryvideo/bears.mpg">Bears Today</a><br>,
<a href="http://cdn-server.com/furryvideo/bunnies.mpg">Funny Bunnies</a><br>
<a href="http://cdn-server.com/furryvideo/mice.mpg">Nice Mice</a><br>
</body> </html>
```

(b)

Fig. 7-46. (a) Pagina Web originală. (b) Aceeaşi pagină după transformare.

Cu un conţinut replicat pe mii de situri în lumea întreagă, există în mod clar un potenţial ridicat pentru îmbunătăţirea performanţelor. Cu toate acestea, pentru o funcţionare bună, trebuie să existe o modalitate prin care să se redirecteze cererea clientului la cel mai apropiat server CDN, preferabil unul aflat la ISP-ul clientului. De asemenea, această redirectare trebuie făcută fără modificarea DNS-ului sau a oricărei alte părţi a infrastructurii standard a Internet-ului. O descriere puţin simplificată despre cum lucrează Akamai, cel mai mare CDN, este oferită în continuare.

Întregul proces începe din momentul în care furnizorul conţinutului trimite CDN-ului situl său Web. Apoi CDN-ul trece fiecare pagină printr-un preprocesor care înlocuieşte toate URL-urile cu unele modificate. Modelul de lucru din spatele acestei strategii este acela că situl Web al furnizorului de conţinut este constituit din multe pagini foarte mici (doar text HTML), dar că aceste pagini au de obicei referinţe către fişiere mari, precum imagini, audio şi video. Paginile HTML modificate sunt păstrate pe serverul furnizorului de conţinut şi sunt aduse în mod obişnuit; doar imaginile, comunicaţiile audio şi video merg pe serverele CDN-ului.

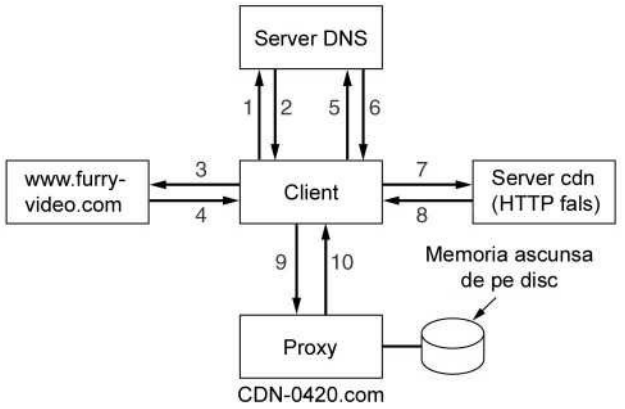
Pentru a vedea cum funcţionează această schemă în realitate, se consideră pagina Web a lui Furry (rom.: îmblănit) Video din fig. 7-46(a). După preprocesare, ea este transformată în fig. 7-46(b) şi plasată pe serverul Furry Video ca *www.furryvideo.com/index.html*.

Când un utilizator introduce ca URL *www.furryvideo.com*, DNS-ul întoarce adresa IP a sitului Furry Video, permiţând ca pagina (HTML) principală să fie adusă în mod obişnuit. Când se face clic pe oricare din hiper-legături, programul de navigare cere DNS-ului să caute *cdn-server.com*, ceea ce acesta chiar face. Programul de navigare trimite apoi o cerere HTTP către această adresă IP, contând pe faptul că va primi înapoi un fişier MPEG.

Aceasta nu se întâmplă deoarece *cdn-server.com* nu găzduieşte nici un conţinut. În schimb, acesta este pe serverul fals de HTTP al CDN-ului. El examinează numele fişierului şi numele serverului pentru a afla care pagină este necesară cărui furnizor de conţinut. De asemenea, examinează adresa IP a cererii sosite şi o caută în baza sa de date pentru a determina unde este posibil să se afle utilizatorul. Cu o astfel de informaţie, el determină care servere CDN de conţinut pot oferi utilizatorului serviciul cel mai bun. Această decizie este dificilă deoarece serverul situat geografic cel mai aproape poate să nu fie cel mai apropiat în termeni de topologie de reţea, iar cel mai apropiat în termeni de topologie de reţea poate fi foarte aglomerat în acel moment. După ce se face o alegere, *cdn-server.com* trimite înapoi un răspuns

cu codul de stare 301 şi un antet *Location* cu URL-ul fişierului pe serverul CDN de conţinut cel mai apropiat de client. Pentru acest exemplu, să presupunem că URL-ul este *www.CDN-0420.com/furryvideo/bears.mpg*. Programul de navigare procesează apoi acest URL în mod normal pentru a obţine fişierul MPEG real.

Paşii urmaţi sunt ilustraţi în fig. 7-47. Primul pas este căutarea *www.furryvideo.com* pentru a obţine adresa lui IP. După aceea, pagina HTML poate fi adusă şi afişată în mod obişnuit. Pagina conţine trei hiper-legături la *cdn-server* [vezi fig. 7-46(b)]. Când, să zicem, este selectată prima, este căutată (pasul 5) şi returnată (pasul 6) adresa sa de DNS. Când o cerere pentru *bears.mpg* este trimisă la *cdn-server* (pasul 7), clientul este înştiinţat că trebuie să se ducă de fapt la *CDN-0420.com* (pasul 8). Când face acest lucru (pasul 9), i se dă fişierul din memoria ascunsă a proxy-ului (pasul 10). Proprietatea care face ca întregul mecanism să funcţioneze este pasul 8, serverul fals de HTTP redirectând clientul la un proxy CDN aflat aproape de client.



1. Caută www.furryvideo.com
2. Este întoarsă adresa IP a lui Furry
3. Pagina HTML este cerută de la Furry
4. Este întoarsă o pagină HTML
5. După selecţia cu mouse-ul, se caută cdn-server.com
6. Este întoarsă adresa IP a lui cdn-server
7. Bears.mpg este cerut de la cdn- server
8. Clientul este redirecţionat către CDN-0420
9. Este cerut fişierul bears.mpg
10. Fişierul bears.mpg este întors din memoria ascunsă.

Fig. 7-47. Paşii în căutarea unui URL când se foloseşte un CDN

Serverul CDN la care este redirectat clientul este în mod tipic un proxy cu o memorie ascunsă preîncărcată cu conţinutul cel mai important. Dacă totuşi cineva cere un fişier care nu este în memoria ascunsă, acesta este adus de la serverul real şi dispus în memoria ascunsă pentru o utilizare ulterioară. Făcând din serverul de conţinut un proxy şi nu o replică completă, CDN are abilitatea de a schimba dimensiunea discului, timpul de preîncărcare şi diverşi parametri de performanţă.

Mai multe despre reţele de livrare a conţinutului găsiţi în (Hull, 2002; şi Rabinovich şi Spatscheck, 2002).

7.3.6 Web-ul fără fir

Există un interes considerabil pentru dispozitivele mici, portabile, capabile să acceseze Web-ul printr-o legătură fără fir. De fapt, primii paşi în această direcţie au fost deja făcuţi. Cu siguranţă că vor fi o mulţime de schimbări în acest domeniu în anii ce vin, dar tot merită să examinăm câteva din idele actuale legate de web-ul fără fir, pentru a vedea unde suntem acum şi încotro ne putem îndrepta. Ne vom concentra asupra primelor două sisteme Web fără fir de scară largă care au spart piaţa: WAP şi i-mode.

WAP-The Wireless Application Protocol (Protocolul pentru aplicaţii fără fir)

O dată ce Internet-ul şi telefoanele mobile au devenit lucruri comune, nu a durat mult până când cuiva i-a venit ideea să le combine într-un telefon mobil cu ecran încorporat pentru acces fără fir la poşta electronică şi la Web. Acel „cineva” a fost consorţiul condus iniţial de Nokia, Ericsson, Motorola şi phone.com (fosta Unwired Planet) şi care acum se laudă cu sute de membri. Sistemul se numeşte **WAP (Wireless Application Protocol –** protocolul pentru aplicaţii fără fir**)**.

Un dispozitiv WAP poate fi un telefon mobil îmbunătăţit, PDA, sau calculator portabil fără servicii pentru voce. Specificaţia le permite pe toate şi multe altele.

Ideea de bază este să se folosească infrastructura digitală fără fir existentă. Utilizatorii pot accesa o poartă (eng.: gateway) WAP prin legătura fără fir şi îi pot trimite cereri de pagini Web. Apoi, poarta controlează memoria ascunsă pentru pagina cerută. Dacă există, o trimite; dacă nu există, o ia de pe Internet-ul cu fir. În esenţă, această înseamnă că WAP 1.0 este un sistem cu comutare de circuite cu o taxă de conectare pe minut relativ mare. Pentru a scurta o poveste lungă, oamenilor nu le-a plăcut sa acceseze Internet-ul pe un ecran mic şi plătind la minut, astfel că WAP-ul a fost un fel de nereuşită (deşi au mai fost şi alte probleme). În orice caz, WAP-ul şi competitorul sau, i-mode (prezentat mai jos), par să conveargă spre o aceeaşi tehnologie, astfel că WAP 2.0 ar mai putea să fie un mare succes. Întrucât WAP 1.0 a fost prima încercare pentru Internet-ul fără fir, merită să fie descris cel puţin pe scurt.

WAP este de fapt o stivă de protocoale pentru accesarea Web-ului, optimizată pentru conexiuni cu o bandă de transfer mică folosind dispozitive fără fir ce au un procesor lent, puţină memorie şi un ecran mic. Aceste cerinţe sunt evident diferite de cele pentru un PC standard de birou, scenariu care duce la nişte diferenţe între protocoale. Nivelurile sunt prezentate în fig. 7-48.

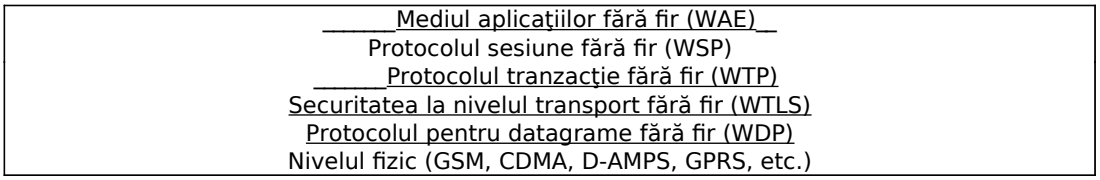

Mediul aplicaţiilor fără fir (WAE)
Protocolul sesiune fără fir (WSP)
Protocolul tranzacţie fără fir (WTP)
Securitatea la nivelul transport fără fir (WTLS)
Protocolul pentru datagrame fără fir (WDP)
Nivelul fizic (GSM, CDMA, D-AMPS, GPRS, etc.)

Fig. 7-48. Stiva de protocoale WAP.

Nivelul cel mai de jos suportă toate sistemele de telefonie mobilă existentă, inclusiv GSM, DAMPS şi CDMA. Rata de transfer pentru WAP 1.0 este de 9600 bps. Deasupra acestora se află protocolul pentru datagrame, **WDP (Wireless Datagram Protocol –** protocolul pentru datagrame fără fir**)**, care este de fapt UDP. Apoi vine un nivel pentru securitate, evident necesar într-un sistem fără fir. WTLS este un subset al SSL-ului de la Netscape, la care ne vom uita în cap. 8. Deasupra acestuia este un nivel tranzacţie sigură sau nesigură, care se ocupă de cereri şi răspunsuri. Acest nivel înlocuieşte TCP, care nu este folosit peste legătura prin aer din motive legate de eficienţă. Apoi vine un nivel sesiune, care este similar cu HTTP/1.1, dar cu câteva restricţii şi extensii pentru motive de optimizare. Deasupra acestuia se află micro-programul de navigare (WAE).

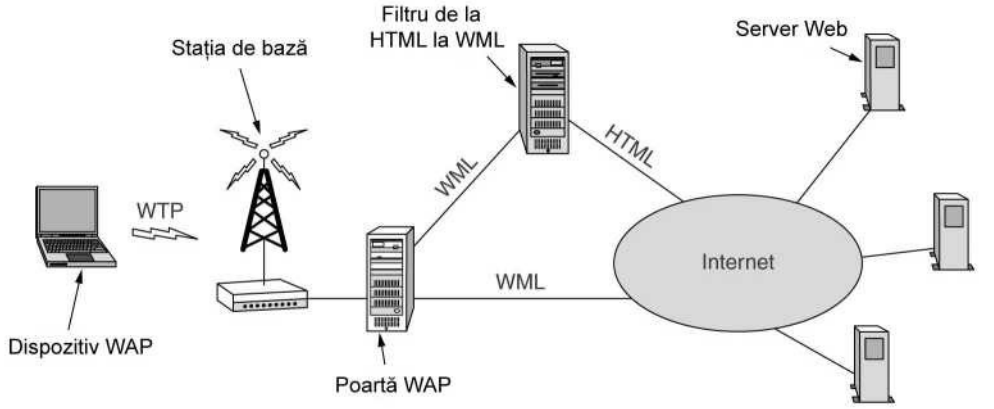


Fig. 7-49. Arhitectura WAP.

În afara costului, celălalt aspect care cu siguranţă a afectat acceptarea WAP-ului este faptul că nu foloseşte HTML. În locul acestuia, nivelul WAE foloseşte un limbaj de marcare numit **WML (Wireless Markup Language –** limbajul de marcare fără fir**)**, care este o aplicaţie a XML. Drept consecinţă, în principiu, un dispozitiv WAP nu poate accesa decât acele pagini care au fost convertite la WML. Oricum, având în vedere că asta restricţionează mult valoarea WAP-ului, arhitectura reclamă un filtru direct de la HTML la WML pentru a creşte mulţimea paginilor disponibile. Arhitectura este ilustrată în fig. 7-49.

Ca să fim corecţi, WAP-ul a fost, probabil, puţin înaintea vremii sale. Când WAP-ul a fost lansat prima dată, XML abia era cunoscut în afara W3C şi astfel presa a anunţat lansarea sa spunând **WAP NU FOLOSEŞTE HTML**. Un titlu mai clar ar fi fost: **WAP DEJA FOLOSEŞTE NOUL STANDARD HTML.** Dar cum răul fusese făcut, a fost greu de reparat şi WAP 1.0 nu a prins niciodată. Vom rediscuta WAP-ul după ce ne vom uita mai întâi la competitorul său major.

I-Mode

În timp ce un consorţiu multi-industrial de companii de telecomunicaţii şi de calculatoare a fost ocupat cu construcţia unui standard deschis folosind cea mai avansată versiune de HTML disponibilă, alte dezvoltări aveau loc în Japonia. Acolo, o japoneză, Mari Matsunaga, a inventat o altă soluţie pentru Web-ul fără fir numită **i-mode (information mode –** modul informaţie**)**. Ea a convins divizia „fără fir” a fostului monopol de telefonie japoneză că ideea sa era corectă şi în februarie 1999 NTT DoCoMo (în traducere literală: Compania Japoneză pentru Telefoane şi Telegraf oriunde te-ai duce) a lansat serviciul în Japonia. În 3 ani a avut peste 35 de milioane de abonaţi japonezi, care puteau accesa peste 40.000 de situri Web speciale i-mode. În plus, a mai făcut ca majoritatea companiilor de telecomunicaţii să saliveze după succesul său financiar, mai ales datorită faptului că WAP nu părea sa ducă nicăieri. Să vedem acum ce este i-mode şi cum funcţionează.

Sistemul i-mode are trei componente de bază: un nou sistem de transmisie, un nou telefon şi un nou limbaj pentru proiectarea paginilor Web. Sistemul de transmisie constă în două reţele separate: reţeaua de telefonie mobilă cu comutare

de circuite existentă (oarecum comparabilă cu D-AMPS) şi o nouă reţea cu comutare de pachete construită în mod special pentru serviciile i-mode. Modul voce foloseşte reţeaua cu comutare de circuite şi este taxat la fiecare minut de conectare. I-mode foloseşte reţeaua cu comutare de pachete şi este întotdeauna activ (la fel ca la ADSL sau la cablu), astfel că nu există taxarea pentru timpul de conectare. În locul acesteia, există o taxă pentru fiecare pachet trimis. Momentan nu este posibil să fie folosite ambele reţele în acelaşi timp.

Telefoanele arată ca nişte telefoane mobile cărora li s-a adăugat un mic ecran. NTT DoCoMo promovează masiv dispozitivele i-mode ca fiind mai degrabă telefoane mobile decât terminale Web fără fir, deşi ele chiar asta sunt. De fapt, probabil că majoritatea clienţilor nici nu sunt conştienţi că sunt conectaţi la Internet. Ei consideră dispozitivele lor i-mode ca fiind telefoane mobile cu facilitaţi sporite. Pentru a păstra acest model de i-mode la nivel de serviciu, telefoanele nu pot fi programate de utilizatori, deşi ele conţin echivalentul unui PC din 1995 şi ar putea probabil rula Windows 95 sau UNIX.

Când telefonul i-mode este pornit, utilizatorului îi este prezentată o listă cu categoriile de servicii aprobate oficial. Sunt mult peste 1000 de servicii grupate în aproximativ 20 de categorii. Fiecare serviciu, care este de fapt un mic sit Web i-mode, este oferit de către o companie independentă. Categoriile importante din meniul oficial includ poşta electronică, ştiri, meteo, sport, jocuri, cumpărături, hărţi, horoscop, distracţie, călătorii, ghiduri regionale, tonuri ale soneriei, reţete, jocuri de noroc, servicii bancare şi cotaţiile bursei. Serviciul este oarecum orientat către adolescenţi şi oameni de 20- 30 de ani, care au tendinţa să se ataşeze de jucăriile electronice, mai ales dacă sunt în culori frumos asortate. Simplul fapt că peste 40 de companii vând tonuri ale soneriei spune ceva. Cea mai populară aplicaţie este poşta electronică, care permite mesaje de până la 500 de octeţi şi din acest motiv este văzută ca o mare îmbunătăţire faţă de SMS (Short Message Service – serviciul de mesaje scurte) care permite mesaje de numai 160 de octeţi. Jocurile sunt şi ele populare.

Sunt de asemenea peste 40.000 de situri Web i-mode, dar ele trebuie accesate mai degrabă scriindu-se URL-ul lor, decât selectându-le dintr-un meniu. Dintr-un punct de vedere, lista oficială este ca un portal Internet care permite altor situri Web să fie accesate prin selecţie în loc să li se scrie URL-ul.

NTT DoCoMo controlează îndeaproape serviciile oficiale. Pentru a fi acceptat pe listă, un serviciu trebuie să îndeplinească o serie de criterii publice. De exemplu, un serviciu nu trebuie să aibă o influenţă negativă asupra societăţii, dicţionarele japonez-englez trebuie să aibă suficiente cuvinte, serviciile cu tonuri pentru sonerie trebuie să adauge frecvent noi tonuri şi nici un sit nu poate să promoveze comportarea vicioasă sau să se reflecte negativ asupra NTT DoCoMo (Frangle, 2002). Cele 40.000 de situri Internet pot face orice vor ele.

Modelul afacerii i-mode este atât de diferit de acela al Internet-ului convenţional încât merită explicat. Taxa pentru abonamentul de bază i-mode este de câţiva dolari pe lună. Cum există o taxă pentru fiecare pachet primit, abonamentul de bază include şi un mic număr de pachete. Ca alternativă, clientul poate opta pentru un abonament cu mai multe pachete gratuite, cu o taxă pe pachet ce scade repede pe

măsură ce trece de la 1 MB pe lună la 10 MB pe lună. Dacă pachetele gratuite sunt folosite până la jumătatea lunii, pot fi cumpărate on-line alte pachete adiţionale.

Pentru a folosi un serviciu trebuie să te abonezi la el, fapt care se realizează printr-o simplă selecţie şi introducerea codului PIN personal. Majoritatea serviciilor oficiale costă în jur de 1-2 pe lună. NTT DoCoMo adaugă taxa la factura de telefon şi transferă 91% celui care oferă serviciul, păstrând 9%. Dacă un serviciu neoficial are 1 milion de clienţi, trebuie sa trimită 1 milion de facturi de (aproximativ) 1$ în fiecare lună. Dacă acel serviciu devine oficial, NTT DoCoMo se ocupă de taxare şi transferă lunar 910.000$ în contul din bancă al serviciului. A nu avea de manipulat note de plată este un mare stimulent pentru ca cineva să devină distribuitor oficial de servicii, ceea ce generează venituri mai mari pentru NTT DoCoMo. De asemenea, fiind oficial ajungi în meniul iniţial, ceea ce face situl tău mult mai uşor de găsit. Factura utilizatorului include convorbirile, taxele de abonamente imode, taxele de abonamente pentru servicii şi pachetele suplimentare.

În ciuda succesului său masiv în Japonia, nu este de loc clar că i-mode va prinde şi în SUA şi Europa. Din unele puncte de vedere, situaţia din Japonia este diferită de aceea din Vest. În primul rând, majoritatea potenţialilor clienţi din Vest (spre exemplu adolescenţii, studenţii şi oamenii de afaceri) au deja un PC cu ecran mare acasă şi aproape sigur o conexiune la Internet de cel puţin 56 Kbps, adesea mult mai rapidă. În Japonia, puţini oameni au PC-uri conectate la Internet acasă, pe de o parte din cauza lipsei de spaţiu, dar şi din cauza taxelor exorbitante ale NTT pentru serviciile de telefonie locală (undeva în jur de 700$ pentru instalarea unei linii şi 1.50$ pe oră pentru convorbiri locale). Pentru majoritatea utilizatorilor, i-mode este singura lor conexiune la Internet.

În al doilea rând, locuitorii din Vest nu sunt obişnuiţi sa plătească 1$ pe lună pentru a accesa situl Web al CNN, 1$ pe lună pentru a accesa situl Yahoo, 1$ pe lună pentru a accesa situl Google şi aşa mai departe, fără să mai menţionăm câţiva dolari pentru fiecare MB descărcat. Majoritatea distribuitorilor de Internet din Vest au acum o taxă fixă pe lună, independentă de utilizarea reală, în mare măsură ca răspuns la cererea clienţilor.

În al treilea rând, pentru mulţi japonezi, perioada de vârf în care folosesc i-mode este perioada în care se deplasează la sau de la serviciu sau şcoală în tren sau în metrou. În Europa, mai puţini oameni se deplasează cu trenul decât în Japonia, iar în SUA abia dacă se deplasează câţiva. Folosirea imode acasă, lângă un calculator cu un monitor de 17 ţoli, conexiune ADSL de 1 Mbps şi toţi megaocteţii gratuiţi, nu se prea justifică. Cu toate acestea, nimeni nu a prezis imensa popularitate a telefoanelor mobile în general, astfel că i-mode mai poate încă să-şi găsească o nişă în Vest.

Aşa cum am menţionat mai sus, telefoanele i-mode folosesc reţeaua cu comutare de circuite existentă pentru voce şi o nouă reţea cu comutare de pachete pentru date. Reţeaua pentru date se bazează pe CDMA şi transmite pachete de 128 de octeţi la 9600 bps. O diagramă a reţelei este dată în fig. 7-50. Telefoanele folosesc **LTP (Lightweight Transport Protocol –** protocol simplificat de transport**)** pe o legătură prin aer până la o poartă pentru conversie de protocoale. Poarta are o conexiune de bandă largă prin fibră optică la serverul i-mode, care este conectat la

toate serviciile. Când utilizatorul selectează un serviciu din meniul oficial, cererea este trimisă serverului i-mode, care ţine majoritatea paginilor în memoria ascunsă pentru a-şi spori performanţa. Cererile pentru situri care nu sunt în meniul oficial ocolesc serverul i-mode şi merg direct pe Internet.

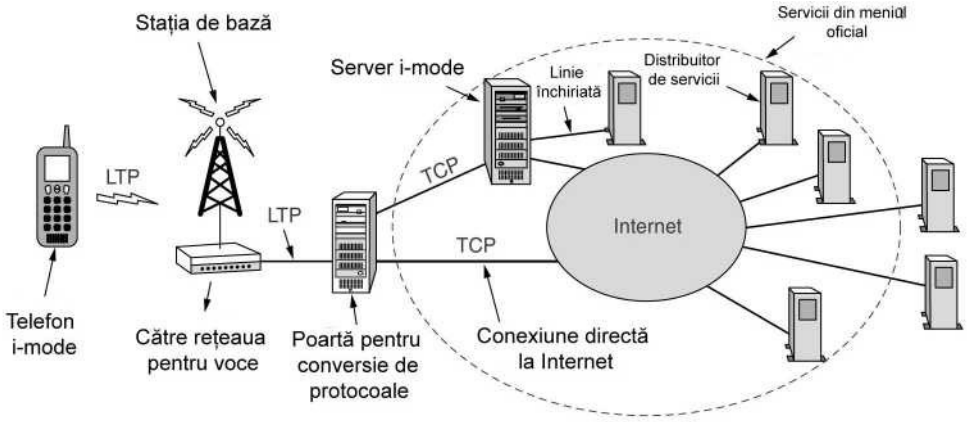


Fig. 7-50. Structura reţelei de date i-mode, arătând protocoalele de transport.

Telefoanele actuale au procesoare care funcţionează la aproximativ 100 MHz, câţiva megaocteţi de memorie ROM rapidă, poate 1 MB RAM şi un ecran mic încorporat. I-mode necesită un ecran de cel puţin 72x94 pixeli, dar unele dispozitive mai mari au chiar 120x160 pixeli. Ecranele au de obicei culori pe 8 biţi, ceea ce permite 256 de culori. Aceasta nu este suficient pentru fotografii, dar este adecvat pentru desenarea de linii şi imagini animate simple. Cum nu există mouse, navigarea pe ecran se face cu săgeţile direcţionale.

Modulul de interacţiune cu utilizatorul
Elemente de intrare | Interpretor cHTML | Java
Coordonator simplu de ferestre
Comunicaţie de reţea
Sistem de operare în timp real

Fig. 7-51. Structura aplicaţiilor i-mode.

Structura aplicaţiilor este prezentată în fig. 7-51. Nivelul cel mai de jos conţine un sistem simplu de operare în timp real pentru controlul echipamentelor. Apoi vine un modul pentru comunicarea pe reţea, folosind protocolul proprietar al NTT DoCoMo, LTP. Deasupra acestuia vine un simplu coordonator de ferestre care se ocupă de text şi de imaginile simple (fişiere GIF). Cu ecranele având doar aproximativ 120x160 de pixeli în cel mai bun caz, nu sunt prea multe de coordonat.

Al patrulea nivel conţine interpretorul de pagini Web (de exemplu, programul de navigare). Imode nu foloseşte întregul HTML, ci numai un subset al acestuia, numit **cHTML (compact HTML –** HTML compact**)**, bazat în mare pe HTML 1.0. Acest nivel permite de asemenea şi aplicaţii ajutătoare şi elemente de intrare, la fel cum fac şi programele de navigare pentru PC-uri. O aplicaţie ajutătoare standard este un

interpretor pentru o versiune puţin modificată a JVM.

La nivelul cel mai înalt se află modulul de interacţiune cu utilizatorul, care controlează comunicaţia cu acesta.

Să ne uităm acum mai în detaliu la cHTML. După cum am menţionat, este aproximativ HTML 1.0, cu câteva omisiuni şi câteva extensii pentru a fi folosit cu telefoane mobile. A fost trimis la W3C pentru standardizare, dar W3C nu a arătat interes pentru el, aşa că probabil va rămâne un produs privat.

Majoritatea etichetelor de bază HTML sunt permise, incluzând aici <html>, <head>, <title>, <body>, <h*n*>, , <ul>, <ol>, <©>, <li>, <br>, <p>, <hr>, <img>, <form> şi <input>. Etichetele <b> şi <i> nu sunt permise.

Eticheta <a> este permisă pentru legarea la alte pagini, dar cu schema adiţională *tel* pentru formarea numerelor de telefon. Dintr-un punct de vedere *tel* este analog cu *mailto.* Când o hiper- legătură folosind schema *mailto* este selectată, programul de navigare deschide un formular pentru a trimite un mesaj electronic către destinaţia marcată în legătură. Când este selectată o hiper-legătură cu schema *tel*, programul de navigare formează numărul de telefon. Spre exemplu, o agendă de telefon poate conţine imagini simple ale unor persoane. Când se selectează una dintre acestea, telefonul îl va suna pe el sau pe ea. RFC 2806 prezintă URL-urile pentru telefoane.

Programul de navigare cHTML este limitat în alte feluri. El nu suportă JavaScript, cadre, foi de stil, culori sau imagini de fundal. De asemenea, nu suportă imagini JPEG, deoarece acestea se decomprimă în prea mult timp. Sunt permise applet-urile Java, dar sunt (în prezent) limitate la 10 KB din cauza vitezei mici de transmisie a legăturii prin aer.

Deşi NIT DoCoMo a eliminat câteva etichete HTML, a şi adăugat unele noi. Eticheta <blink> face ca textul să se afişeze şi apoi să dispară. Deşi pare ciudat să interzici eticheta <b> (motivând că siturile Web nu ar trebui să se ocupe de prezentarea conţinutului) şi apoi să adaugi <blink> care nu se referă decât la prezentarea conţinutului, asta au făcut. O altă etichetă nouă este <marquee>, care îşi deplasează conţinutul pe ecran precum un monitor de bursă.

Un element nou este atributul *align* al etichetei <br>. Este necesar, deoarece pentru un ecran ce are de obicei 6 rânduri de câte 16 caractere, există un mare pericol ca rândurile să fie rupte în două, ca în fig. 7-52(a). *Align* ajută la reducerea acestei probleme şi conduce la ceva ce seamănă mai degrabă cu fig. 7-52(b). Este interesant să notăm că limba japoneză nu suferă când cuvintele sunt rupte între linii. Pentru textul kanji, ecranul este împărţit într-un caroiaj dreptunghiular de celule de dimensiune 9x10 pixeli sau 12x12 pixeli, în funcţie de caracterele suportate. Fiecare celulă conţine exact un caracter kanji, care este echivalentul unui cuvânt în engleză. Despărţirea mai multor cuvinte pe mai multe linii este întotdeauna admisă în japoneză.

Veni vremea când morsa spuse că v a vorbi despre m ulte lucruri. De spre pantofi şi
Veni vremea când morsa spuse că va vorbi despre multe lucruri. Despre pantofi
(a)
(b)

Fig. 7-52. Lewis Caroll încape într-un ecran de 16x6.

Deşi limba japoneză are zeci de mii de kanji, NTT DoCoMo a inventat 166 noi, numite **emoji**, cu un factor de amuzament ridicat – de fapt, nişte pictograme precum zâmbitorii din fig. 7-6. Acestea includ simboluri pentru semnele astrologice, bere, hamburger, parc de amuzament, zi de naştere, telefon mobil, câine, pisica, Crăciun, inimă rănită, sărut, stare de spirit, somnolenţă, şi desigur, unul semnificând ceva simpatic.

Un alt nou element este posibilitatea de a permite utilizatorilor să selecteze hiper-legături folosind tastatura, proprietate cu siguranţă importantă pentru un calculator fără mouse. Un exemplu despre cum este folosit acest atribut este prezentat în fişierul cHTML din fig. 7-53.

```
<html>
<body>
<h1> Selectează o opţiune </h1>
<a href="messages.chtml" accesskey="1"> Verifică poşta vocală </a> <br>
<a href="mail.chtml" accesskey="2"> Verifică poşta electronică </a> <br>
<a href="games.chtml" accesskey="3"> Joacă un joc </a> </body> </html>
```

Fig. 7-53. Un exemplu de fişier cHTML.

Deşi partea client este oarecum limitată, serverul i-mode este un calculator de-a dreptul răsunător, cu toate soneriile şi fluierele obişnuite. El suportă CGI, Perl, PHP, JSP, ASP şi tot ceea ce serverele Web suportă de obicei.

Caracteristica	**WAP**	**I-mode**
Ce este	Stivă de protocoale	Serviciu
Dispozitiv	Telefon, PDA, calculator portabil	Telefon
Tipul de acces	Prin telefon	Tot timpul activ
Reţeaua de suport	Cu comutare de circuite	Doua: circuite + pachete
Rata de transfer	9600 bps	9600 bps
Ecranul	Monocrom	Color
Limbajul de marcare	WML (aplicaţie XML)	cHTML
Limbajul script	WMLScript	Nici unul
Taxarea utilizatorilor	Pe minut	Pe pachet
Plata pentru	Cu cartea de credit	Pe factura de telefon
Pictograme	Nu	Da
Standardizare	Standard deschis al forumului	Proprietar NTT DoCoMo
Locul de utilizare	Europa, Japonia	Japonia
Utilizatorul tipic	Omul de afaceri	Adolescentă

O comparaţie rapidă între WAP şi i-mode aşa cum au fost ele implementate în

Fig. 7-54. O comparaţie între prima generaţie de WAP şi i-mode.

sistemele de prima generaţie este dată în fig. 7-54. În timp ce câteva diferenţe pot părea mici, adesea ele sunt importante. Spre exemplu, cei de 15 ani nu au cărţi de credit, astfel că posibilitatea de a cumpăra lucruri prin comerţul electronic şi de a fi taxaţi abia când primesc nota de plată la telefon reprezintă un stimulent pentru interesul lor asupra sistemului.

Pentru alte informaţii despre i-mode citiţi (Frengle, 2002; şi Vacca, 2002).

Web-ul fără fir de generaţia a doua

WAP 1.0, bazat pe standarde recunoscute internaţional, trebuia să fie o unealtă importantă pentru oamenii cu afaceri în derulare. A eşuat. I-mode a fost o jucărie electronică pentru adolescenţii japonezi folosind numai elemente proprietare. A fost un mare succes. Ce s-a întâmplat după asta? Fiecare parte a învăţat câte ceva din Web-ul fără fir de primă generaţie. Consorţiul WAP a învăţat că important este conţinutul. Să nu ai un număr mare de situri Web care îţi înţeleg limbajul de marcare este fatal. NTT DoCoMo a învăţat că un sistem închis, proprietar, strâns legat de dispozitive mici şi de cultura japoneză nu este un bun produs pentru export. Concluzia pe care ambele părţi au tras-o este că pentru a convinge un număr mare de situri Web să-şi pună conţinutul în formatul tău, trebuie să ai un limbaj de marcare deschis, stabil şi care este universal acceptat. Războaiele asupra formatelor nu sunt bune pentru progres.

Ambele servicii sunt aproape de a intra în cea de-a doua generaţie a tehnologiei Web fără fir. WAP 2.0 a venit primul, aşa că îl vom folosi pe acesta ca exemplu. WAP 1.0 avea unele lucruri bune şi acestea au fost continuate. Unul dintre acestea este că WAP poate folosi o mulţime de reţele diferite. Prima generaţie folosea reţele cu comutare de circuite, dar reţelele cu comutare de pachete au fost întotdeauna o alternativă şi încă mai sunt. Sistemele de a doua generaţie vor folosi probabil comutarea de pachete, spre exemplu, GPRS-ul. Un altul este că WAP a fost iniţial proiectat pentru a suporta o mare varietate de dispozitive, de la telefoane mobile până la calculatoare portabile puternice, şi încă mai este.

WAP 2.0 are şi câteva caracteristici noi. Cele mai importante sunt:

1. Modelul de livrare a paginilor (push), alături de cel de cerere (pull).
2. Suport pentru integrarea telefoniei în aplicaţii.
3. Mesageria multimedia.
4. Includerea a 264 de pictograme.
5. Interfaţa cu un dispozitiv de memorare.
6. Suport pentru plug-în-uri în browser.

Modelul de cerere este bine cunoscut: clientul cere o pagină şi o primeşte. Modelul de livrare (push) suportă livrarea datelor fără a fi cerute, precum ţinerea la curent cu cotaţiile la bursă sau alertele de trafic.

Vocea şi datele încep să se contopească, iar WAP 2.0 le suportă într-o mulţime de moduri. Am văzut un astfel exemplu mai devreme, la capabilitatea i-mode-ului de a lega o iconiţă sau un text de pe ecran cu un număr de telefon ce trebuie format. Odată cu poşta electronică şi telefonia este suportată şi mesageria multimedia.

Marea popularitate a emoji-ului i-mode a stimulat consorţiul WAP să inventeze

264 emoji proprii. Categoriile includ animale, utilaje casnice, îmbrăcăminte, emoţii, mâncăruri, corpul uman, genuri, hărţi, muzică, plante, sporturi, timp, unelte, vehicule, arme şi meteo. Destul de interesant este faptul că standardul pur şi simplu numeşte fiecare pictogramă; nu dă harta de pixeli reală, probabil de teamă că reprezentarea într-o cultură a „somnolenţei" sau a „îmbrăţişării" să nu insulte altă cultură. I-mode nu a avut această problemă fiind îndreptat către o singură ţară.

A oferi o interfaţă de stocare nu înseamnă că fiecare telefon cu WAP 2.0 va veni cu un mare disc fix. Memoria ROM rapidă este, de asemenea, un dispozitiv de stocare. O cameră fără fir cu facilităţi WAP ar putea folosi memoria ROM rapidă pentru stocarea temporară a imaginii înainte de a transmite cele mai bune cadre pe Internet.

În fine, plug-în-urile pot extinde capabilităţile programului de navigare. Este oferit, de asemenea, un limbaj scriptic.

Mai multe diferenţe tehnice sunt de asemenea prezente în WAP 2.0. Două dintre cele mai importante se referă la stiva de protocoale şi la limbajul de marcare. WAP 2.0 continuă să suporte vechea stivă de protocoale din fig. 7-48, dar de asemenea suportă şi stiva standard a Internet-ului cu TCP şi ©/1.0. Cu toate acestea, patru schimbări minore (dar compatibile) au fost aduse TCP-ului (pentru a simplifica codul): (1) Folosirea unei ferestre fixe de 64 KB, (2) lipsa unui start lent, (3) MTU-ul maxim de 1500 de octeţi şi (4) un algoritm de retransmisie uşor modificat. TLS este protocolul pentru securitatea la nivel transport standardizat de IETF; îl vom examina în Cap. 8. Mai multe dispozitive iniţiale vor conţine probabil ambele stive, cum se arată în fig. 7-55.

Fig. 7-55. WAP 2.0 suportă două stive de protocoale.

Cealaltă diferenţă tehnică faţă de WAP 1.0 este limbajul de marcare. WAP 2.0 suportă XHTML Basic, care este potrivit pentru dispozitivele mici fără fir. Întrucât NTT DoCoMo a fost de asemenea de acord să suporte acest subset, proiectanţii de situri Web pot folosi acest format ştiind că paginile lor vor funcţiona atât pe Internet-ul fix cât şi pe toate dispozitivele fără fir. Aceste decizii vor încheia războaiele legate de formatul limbajului de marcare care au împiedicat dezvoltarea industriei Web fără fir.

Modulul	Obligatoriu?	Funcţia	Exemple de etichete
Structură	Da	Structura	body, head, html, title
Text	Da	Informaţii	br, code, dfn, em, h*n*, kbd, p, strong
Hiper-text	Da	Hiper-legături	a
Liste	Da	Liste de articole	dl, dt, dd, ol, ul, li

Formulare	Nu	Formulare de	form, input, label, option, textarea
Tabele	Nu	Tabele	caption, table, td, th, tr
Imagini	Nu	Imagini	img
Obiecte	Nu	Applet-uri, harţi, etc.	object, param
Meta-informaţii	Nu	Informaţii	meta
Legături	Nu	Similar cu <a>	link
Bază	Nu	URL-ul de start	base

Fig. 7-56. Modulele şi etichetele din XHTML Basic.

Câteva cuvinte despre XHTML Basic sunt poate necesare. Acesta este gândit pentru telefoane mobile, televiziune, PDA-uri, dispozitive pentru vânzarea automată, pagere, maşini, jocuri mecanice şi chiar ceasuri. Din acest motiv nu suportă foi de stil, scripturi sau cadre, însă cunoaşte majoritatea eticheteloor standard. Acestea sunt grupate în 11 module. Unele sunt obligatorii; altele sunt opţionale. Toate sunt definite în XML. Modulele şi câteva exemple de etichete sunt listate în fig. 7-56. Nu baleiat toate exemplele de etichete, însă mai multe informaţii se pot găsi la *www.w3.org*.

În ciuda înţelegerii asupra folosirii XHTML Basic, o ameninţare pândeşte WAP şi i-mode: 802.11. A doua generaţie a Web-ului fără fir ar trebui să funcţioneze la 384 Kbps, mult mai mult decât cei 9600 bps ai primei generaţii, dar şi mult mai puţin decât cei 11 Mbps sau 54 Mbps oferiţi de 802.11. Fireşte, 802.11 nu este omniprezent, dar pe măsură ce mai multe restaurante, hoteluri, magazine, companii, aeroporturi, staţii de autobuz, muzee, universităţi, spitale şi alte organizaţii decid să instaleze staţii de bază pentru angajaţii şi clienţii lor, se va ajunge probabil la o suficientă acoperire în zonele urbane astfel încât oamenii să dorească să meargă pentru o cafea sau pentru a trimite un mesaj electronic la o braserie aflată la câteva blocuri distanţă, dar cu 802.11 instalat. Firmele pot adăuga automat embleme 802.11 alături de emblemele care arată ce cărţi de credit acceptă şi asta din acelaşi motiv: pentru a atrage clienţi. Hărţile oraşelor (fireşte, descărcabile) pot marca zonele acoperite cu verde şi pe cele neconectate cu roşu, astfel ca oamenii să se poată deplasa de la o staţie de bază la alta, aşa cum nomazii se mutau de la o oază la alta în deşert.

Deşi braseriile pot instala repede staţii de bază 802.11, fermierilor probabil că nu le va fi la fel de uşor, deci acoperirea va fi zonală şi limitată la zonele centrale ale oraşelor, din cauza răspândirii limitate a semnalului 802.11 (câteva sute de metri în cel mai bun caz). Aceasta poate duce la dispozitive fără fir cu două moduri, care folosesc 802.11 dacă pot prinde un semnal şi recurg la WAP dacă nu.

7.4 MULTIMEDIA

Deşi Web-ul fără fir este o tehnologie nouă şi incitantă, ea nu este singura. Pentru mulţi, multimedia reprezintă sarea şi piperul reţelelor de calculatoare. Minţile ascuţite văd imense provocări tehnice în furnizarea de video (interactiv) la cerere în fiecare casă. Gulerele albe văd un profit imens în acestea. Întrucât

multimedia necesită o lăţime de bandă mare, este destul de greu să fie făcută să funcţioneze prin conexiuni fixe. Chiar şi calitatea video VHS prin legătura fără fir este la distanţă de câţiva ani, aşa că discuţia noastră se va axa asupra sistemelor conectate.

Literal, multimedia înseamnă două sau mai multe media. Dacă editorul acestei cărţi voia să se alăture interesului la modă despre multimedia, el putea anunţa că lucrarea foloseşte tehnologia multimedia. În fond, aceasta conţine două media: textul şi grafica (desenele). Cu toate acestea, atunci când majoritatea oamenilor se referă la multimedia, de fapt ei se referă la combinarea între două sau mai multe **media continue** (continuous media), adică media care trebuie să se desfăşoare într-un interval bine definit, de obicei folosind interacţiunea cu utilizatorul. În practică, cele două media sunt audio şi video, adică sunete plus filme.

Oricum, mulţi vorbesc adesea despre sunetele audio pure, precum telefonia prin Internet sau ra- dio-ul prin Internet, ca şi cum ar fi tot multimedia, ceea ce cu siguranţă nu sunt. De fapt, un termen mai bun ar fi **fluxuri media** (streaming media), dar vom urma mulţimea şi vom considera sunetele audio în timp real ca fiind tot multimedia. În secţiunile următoare vom examina modul în care calcu

latoarele procesează datele audio şi video, cum le comprimă, şi câteva aplicaţii pentru reţele ale acestor tehnologii. Pentru o tratare cuprinzătoare (trei volume) a datelor multimedia în reţele, citiţi (Steinmetz şi Nahrstedt, 2002; Steinmetz şi Nahrstedt, 2003a; şi Steinmetz şi Nahrstedt, 2003b).

7.4.1 Introducere în sunetele digitale

O undă (sunet) audio este o undă cu o dimensiune acustică (presiune). Atunci când o undă acustică intră în ureche, pavilionul vibrează, făcând ca oasele urechii interne să vibreze o dată cu el, transmiţând vibraţii nervoase creierului. Aceste vibraţii sunt percepute drept sunete de către ascultător. Într-un mod similar, atunci când o undă acustică loveşte un microfon, acesta generează un semnal electric, reprezentând amplitudinea sunetului ca funcţie de timp. Reprezentarea, procesarea, memorarea, şi transmisia acestor semnale audio constituie p ărţile majore ale studiului sistemelor multimedia.

Intervalul de frecvenţă pentru urechea umană este cuprins între 20 Hz şi 20.000 Hz, deşi unele animale, în special câinii, pot percepe frecvenţe mai înalte. Urechea percepe sunetele în mod loga- ritmic, aşa încât raportul a două semnale cu puterile A şi B este exprimat convenţional în **dB (decibeli)** în concordanţă cu formula:

$$\text{dB} = 10 \log_{10}(A/B)$$

Dacă definim limita inferioară de audibilitate (o presiune de circa 0,0003 dyne/cm^2) pentru o undă sinusoidală de 1 KHz ca 0 dB, o conversaţie obişnuită este în jur de 50 dB şi pragul de durere este în jur de 120 dB, un interval dinamic cu un factor de 1 milion. Pentru a evita orice confuzie, A şi B de mai sus sunt numite *amplitudini.* Dacă trebuie să folosim nivelul puterii, care este proporţional cu pătratul amplitudinii, coeficientul logaritmului va fi 10, nu 20.

Urechea este surprinzător de sensibilă la variaţii ale sunetului care durează doar câteva milise- cunde. Ochiul, în schimb, nu poate observa schimbările de nivel ridicat care durează doar câteva milisecunde. Rezultatul acestei observaţii constă în faptul că o variaţie de numai câteva milisecunde din timpul unei transmisii multimedia afectează calitatea sunetului perceput mai mult decât afectează calitatea imaginii percepute.

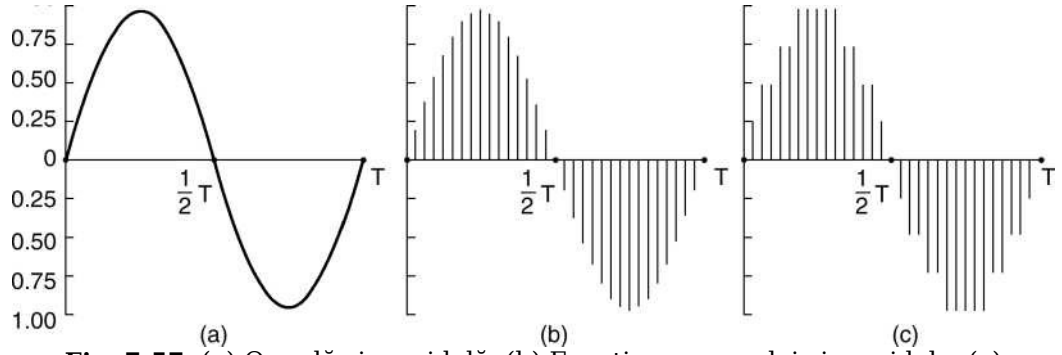


Fig. 7-57. (a) O undă sinusoidală. (b) Eşantionarea undei sinusoidale. (c) Cuantificarea eşantioanelor pe 4 biţi.

Undele audio pot fi convertite în formă numerică de un **ADC (Analog Digital Convertor –** convertor analog numeric**)**. Un ADC primeşte o tensiune electrică la intrare şi generează un număr binar la ieşire. În fig. 7-57(a) se prezintă un exemplu al unei unde sinusoidale. Pentru reprezentarea acestui semnal în formă digitizată, putem să-l eşantionăm la fiecare *AT* secunde, aşa cum puteţi observa prin înălţimea barelor în fig. 7-57(b). Dacă o undă sonică nu este o undă pur sinusoidală, ci o superpoziţie liniară de unde sinusoidale, unde cea mai mare componentă a frecvenţei prezente este *f,* atunci teorema lui Nyquist (vezi cap. 2) spune că este suficient să se eşantioneze la frecvenţa *2f.* Eşantionând mai des, nu se obţine nimic în plus, deoarece frecvenţele mai înalte pe care discretiza- rea le poate detecta nu sunt prezente.

Eşantioanele digitale nu sunt niciodată exacte. Eşantioanele pe 3-biţi din fig. 7-57© permit doar opt valori, de la -l,00 la +1,00 în paşi de 0,25. Un eşantion pe 8-biţi permite 256 valori distincte. Un eşantion pe 16-biţi admite 65.536 valori distincte. Eroarea introdusă de numărul finit de biţi de eşantionare se numeşte **zgomot de cuantizare** (quantization noise). Dacă este foarte mare, urechea îl detectează.

Două exemple bine cunoscute de sunete eşantionate sunt telefonul şi compact discurile audio. Modularea impulsului în cod, folosită în sistemul telefonic, utilizează eşantioane pe 8 biţi de 8000 de ori pe secundă. În America de Nord şi Japonia, 7 biţi sunt pentru date, iar unul pentru control; în Europa toţi cei 8 biţi sunt pentru date. Acest sistem furnizează o viteză a datelor de 56.000 bps sau 64.000 bps. Cu doar 8000 de eşantioane/sec, frecvenţele peste 4 KHz sunt pierdute.

CD-urile audio sunt digitale, cu o rată de eşantionare de 44.100 eşantioane/sec, suficientă pentru a putea capta frecvenţele până la 22.050 Hz, care sunt bune

pentru oameni, dar rele pentru iubitorii canini de muzică. Eşantioanele au fiecare 16 biţi şi sunt liniare în domeniul amplitudinii. Observaţi că eşantioanele pe 16-biţi permit doar 65.536 valori distincte, chiar dacă limita dinamică a urechii este de 1 milion atunci când se măsoară în unităţi de cel mai mic sunet audibil. Astfel, folosind doar 16 biţi pe eşantion, se introduce un zgomot de cuantizare (deşi întreaga rată dinamică nu este acoperită – CD-urile se presupune că nu rănesc). Cu 44.100 eşantioane/sec pe 16 biţi fiecare, un CD audio are nevoie de o lărgime de bandă de 705,6 Kbps pentru mono şi de 1,411 Mbps pentru stereo. Deşi acesta este mai scăzut decât necesită video-ul (vezi mai jos), poate acapara aproape un canal întreg T1 pentru transmiterea necomprimată a sunetului stereo de calitate pentru CD în timp real.

Sunetele digitizate pot fi procesate uşor de calculatoare, prin programe. Există zeci de programe pentru calculatoarele personale care permit utilizatorilor să înregistreze, să afişeze, să editeze, să amestece şi să memoreze undele sonore de la surse multiple. Practic, toate înregistrările, editările de sunete profesioniste sunt în prezent digitale.

Muzica este desigur un caz special de audio, dar unul important. Alt caz important este discursul. Discursul uman tinde să fie între limitele de 600 Hz şi 6000 Hz. Discursul este format din vocale şi consoane, care au proprietăţi diferite. Vocalele sunt produse atunci când coardele vocale sunt neob- strucţionate, producând rezonanţe ale căror frecvenţe fundamentale depind de dimensiunea şi forma sistemului vocal şi de poziţia limbii vorbitorului şi a gurii. Aceste sunete sunt aproape periodice pentru intervale în jur de 30 ms. Consoanele sunt produse atunci când coarda vocală este parţial blocată. Aceste sunete sunt mai puţin regulate decât vocalele.

Câteva sisteme de generare şi transmisie de voce pot folosi modelele de sisteme vocale pentru a reduce vocea la câţiva parametri (de exemplu, mărimile şi formele diferitelor cavităţi), mai degrabă decât de a eşantiona forma de undă pentru voce. Oricum, modul de funcţionare al acestor codoare de voce depăşeşte domeniul acestei cărţi.

7.4.2 Compresia audio

Pentru a obţine calitatea unui CD audio este necesară o lăţime a benzii de transmisie de 1.411 Mbps, după cum tocmai am văzut. Evident, pentru a face practică transmisia pe Internet este necesară o compresie substanţială. Mulţi algoritmi de compresie audio au fost dezvoltaţi din acest motiv. Probabil cel mai popular dintre ele este MPEG audio, care are trei nivele (variante), dintre care **MP3 (MPEG audio layer 3 –** MPEG audio de nivelul 3**)** este cel mai puternic şi mai cunoscut. Pe Internet sunt disponibile mari cantităţi de muzică în format MP3, nu toate legale, fapt care a generat numeroase procese venite din partea artiştilor şi a proprietarilor de drepturi de autor. MP3 aparţine părţii audio a standardului MPEG pentru compresia video. Vom discuta despre compresia video mai târziu în acest capitol; să vedem acum compresia audio.

Compresia audio poate fi făcută în două moduri. Prin **codificarea formei de**

undă (waveform coding)**,** semnalul este transformat (matematic) Fourier în componentele sale în frecvenţă. Fig. 2-1(a) arată un exemplu de funcţie de timp şi amplitudinile sale Fourier. Apoi, amplitudinea fiecărei componente este codificată minimal. Scopul este de a reproduce exact forma de undă la celălalt capăt folosind cât mai puţini biţi cu putinţă.

Cealaltă metodă, **codificarea perceptivă** (eng. Perceptual coding), exploatează unele caracteristici ale sistemului auditiv uman pentru a codifica semnalul astfel încât să sune la fel pentru ascultătorul uman, chiar dacă arată destul de diferit pe osciloscop. Codificarea perceptivă se bazează pe ştiinţa **psihoacusticii –** modul în care oamenii percep sunetul. MP3 se bazează pe codificarea perceptivă.

Proprietatea de bază a codificării perceptive este că unele sunete pot **masca** alte sunete. Imagi- naţi-vă că transmiteţi în direct un concert de flaut într-o zi călduroasă de vară. Apoi, dintr-o dată, un grup de muncitori din apropiere îşi pornesc ciocanele pneumatice şi încep să distrugă strada. Nimeni nu mai poate auzi flautul. Sunetele sale au fost mascate de ciocanele pneumatice. Din punct de vedere al transmisiei, acum este suficient să codificăm doar banda de frecvenţe folosită de ciocanele pneumatice, deoarece ascultătorii oricum nu pot auzi flautele. Acest procedeu se numeşte **mascarea frecvenţei** (frequency masking) – proprietatea unui sunet de volum înalt dintr-o bandă de frecvenţă de a ascunde un sunet mai slab dintr-o altă bandă de frecvenţă şi care s-ar fi auzit în absenţa sunetului de volum înalt. De fapt, chiar şi după ce ciocanele pneumatice încetează, flautul nu se va putea auzi pentru o scurtă perioadă de timp, deoarece urechile opresc amplificarea sunetelor când acestea încep şi au nevoie de o perioad ă de timp finită pentru a o reporni. Acest efect se numeşte **mascare temporală** (temporal masking).

Pentru a cuantifica aceste efecte, imaginaţi-vă experimentul 1. O persoană dintr-o cameră în care este linişte îşi pune căştile conectate la placa de sunet a unui calculator. Calculatorul generează o undă sinusoidală nedistorsionată la 100 Hz la o putere mică iniţial dar crescătoare în timp. Persoana este instruită să apese pe o tastă când aude sunetul. Calculatorul înregistrează nivelul curent al puterii şi repetă experimentul la 200 Hz, 300 Hz şi toate celelalte frecvenţe până la limita auzului uman. Când se face o medie pentru mai mulţi oameni, un grafic al înregistrărilor puterii necesare unui sunet pentru a fi auzit arată ca cel din fig. 7-58(a). O consecinţă directă a acestei curbe este că nu trebuie să codificăm frecvenţele a căror putere se află sub pragul auzului. De exemplu, dacă puterea la 100 Hz ar fi fost 20 dB în fig. 7-58(a), ar fi putut fi omisă din sunetul final fără o pierdere sesizabilă de calitate, deoarece 20 dB la 100 Hz sunt sub nivelul de audibilitate.

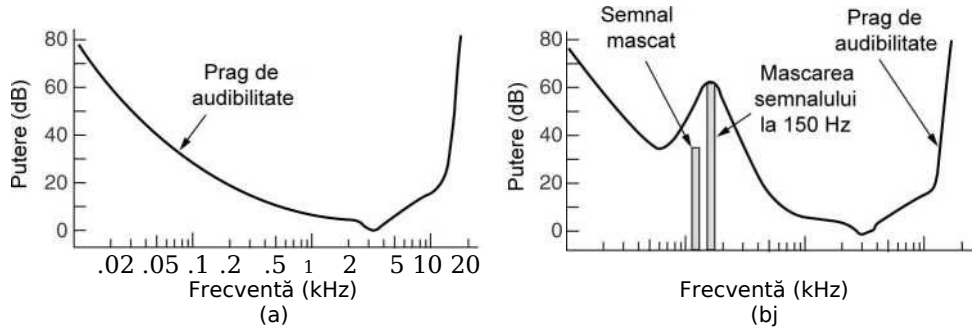


Fig. 7-58. (a) Pragul de audibilitate ca funcţie de frecvenţă. (b) Efectul de mascare.

Acum imaginaţi-vă experimentul 2. Calculatorul rulează din nou experimentul 1, dar de această dată cu o undă sinusoidală de amplitudine constantă şi frecvenţa 150 Hz, suprapusă peste frecvenţa de test. Descoperim că pragul de audibilitate pentru frecvenţe de aproximativ 150 Hz este ridicat, aşa cum arată şi fig. 7-58(b).

Consecinţa acestei noi observaţii este că, pe măsură ce descoperim care semnale sunt mascate de alte semnale din benzi de frecvenţă apropiate, putem omite mai multe frecvenţe din semnalul codificat, economisind biţi. În fig. 7-58, semnalul de 125 Hz poate fi în întregime omis din semnalul de ieşire şi nimeni nu va putea sesiza diferenţa. Chiar şi după ce un semnal puternic dintr-o bandă de frecvenţă se opreşte, cunoaşterea proprietăţilor sale de mascare temporală ne permit să continuăm să omitem frecvenţele mascate pentru un anumit interval de timp până când urechea îşi revine. Elementul de bază al MP3 este transformarea Fourier a sunetului pentru a obţine puterea pentru fiecare frecvenţă şi apoi transmiterea numai a frecvenţelor nemascate, codificându-le în cât mai puţini biţi cu putinţă.

Cunoscând aceste informaţii, putem acum să vedem cum se face codificarea. Compresia audio se face eşantionând forma de undă la 32 KHz, 44.1 KHz sau 48 KHz. Eşantionarea se poate face pe unul sau două canale, în una din cele patru configuraţii:

1. Monofonic (un singur flux de intrare).
2. Dual monofonic (spre exemplu, o coloană sonoră în engleză şi una în japoneză).
3. Stereo disjunct (fiecare canal comprimat separat).
4. Stereo unit (se exploatează la maxim redundanţa inter-canale).

Mai întâi se alege rata biţilor de la ieşire. MP3 poate comprima un CD rock'n'roll stereo până la minim 96 Kbps cu pierderi de calitate greu perceptibile, chiar şi pentru fanii rock'n'roll care nu au probleme cu auzul. Pentru un concert de pian sunt necesari cel puţin 128 Kbps. Acestea diferă deoarece raportul semnal-zgomot pentru rock'n'roll este mult mai mare faţă de cel al unui concert de pian (cel puţin din punct de vedere ingineresc). Este de asemenea posibil să alegem rate de ieşire mai mici acceptând o pierdere în calitate.

Apoi eşantioanele sunt procesate în grupuri de 1152 (lungi de aproximativ 26

ms). Fiecare grup este întâi trecut prin 32 de filtre digitale pentru a obţine 32 de benzi de frecvenţă. În acelaşi timp, intrarea

este dată unui model psiho-acustic care determină frecvenţele mascate. Apoi, fiecare din cele 32 de benzi de frecvenţă este transformată în continuare pentru a obţine o rezoluţie spectrală mai fină.

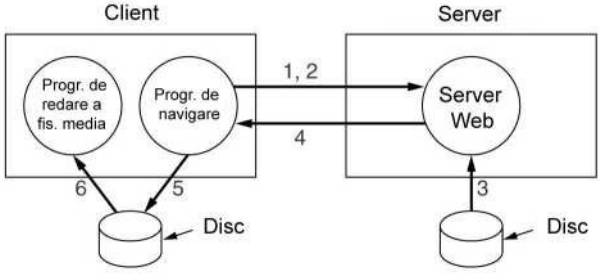
În faza următoare, bugetul de biţi disponibil este împărţit între benzi, cât mai mulţi biţi alocaţi pentru benzile cu puterea spectrală nemascată mai mare, cât mai puţini biţi alocaţi pentru benzile nemascate cu puterea spectrală mai mică şi nici un bit alocat pentru benzile mascate. În final, biţii sunt codificaţi folosind codificarea Huffman, care asignează coduri scurte numerelor care apar frecvent şi coduri lungi celor care apar rar.

În realitate mai sunt multe de spus. Se folosesc de asemenea multe tehnici pentru reducerea zgomotelor, antialiasing şi exploatarea redundanţei inter-canale, dacă este posibil, dar acestea sunt în afara domeniului acestei cărţi. O introducere matematică mai formală în această operaţie este dată în (Pan, 1995).

7.4.3 Fluxuri audio

Să trecem acum de la tehnologia audio digitală la trei dintre aplicaţiile sale pentru reţele. Prima este fluxul audio, adică ascultarea sunetelor pe Internet. Aceasta se numeşte de asemenea şi muzică la cerere. Celelalte două sunt radio-ul prin Internet şi respectiv vocea peste IP.

Internet-ul este plin de situri Web cu muzică, multe dintre ele listând titluri de cântece pe care utilizatorii le pot selecta cu ajutorul mouse-ului pentru le asculta. Unele dintre acestea sunt situri gratuite (spre exemplu formaţiile noi care încearcă să îşi facă publicitate); altele necesită o plată pentru muzică, deşi ele oferă de asemenea mostre gratuite (spre exemplu primele 15 secunde dintr-un cântec). Metoda cea mai rapidă de a asculta muzica este ilustrată în fig. 7-59.



1. Stabilirea conexiunii TCP
2. Trimiterea cererii HTTP GET
3. Serverul preia fişierul de pe disc
4. Fişierul este trimis înapoi programului de navigare
5. Programul de navigare scrie fişierul pe disc
6. Programul pentru redarea fişierelor media preia fişierul bloc după bloc şi îl redă

Fig. 7-59. O metodă directă pentru a implementa muzica selecţionabilă de pe o pagină Web.

Procesul începe când utilizatorul selectează cu mouse-ul un cântec. Apoi intră în acţiune programul de navigare. Primul său pas este stabilirea unei conexiuni TCP cu serverul Web pe care există o hiper-legătură la cântec. Pasul 2 este trimiterea

unei cereri *GET* în HTTP pentru a cere cântecul. Ulterior (paşii 3 şi 4), serverul citeşte cântecul (care este doar un fişier în formatul MP3 sau vreun alt format) de pe disc şi îl trimite înapoi programului de navigare. Dacă fişierul este mai mare decât memoria serverului, acesta poate aduce şi trimite melodia în blocuri.

Folosind tipul MIME, de exemplu, *audiolmp3,* (sau extensia fişierului), programul de navigare caută să vadă cum ar trebui livrat fişierul. De obicei există o aplicaţie ajutătoare asociată cu acest tip de fişier, precum RealOne Player, Windows Media Player, sau Winamp. Întrucât în mod obişnuit programul de navigare comunică cu o aplicaţie ajutătoare printr-un fişier auxiliar, acesta va salva mai întâi întregul fişier de muzică pe disc ca un fişier auxiliar (pasul 5). Apoi, va porni programul de re-

dare a fişierului căruia îi va trimite numele fişierului auxiliar de pe disc. La pasul 6, programul pentru redarea fişierelor media va începe să încarce şi să reproducă muzica, bloc după bloc.

În principiu, această metodă este în întregime corectă şi va produce muzică. Singura problemă este că trebuie trimis prin reţea tot cântecul, înainte ca muzica să înceapă. În cazul în care cântecul are 4 MB (o dimensiune tipică pentru un cântec MP3) şi modemul este de 56 Kbps, utilizatorul va avea parte de aproape 10 minute de linişte până când cântecul este descărcat. Nu toţi îndrăgostiţii de muzică agreează această idee. Mai ales având în vedere că următorul cântec va începe tot după 10 minute de descărcare, iar cel de după acesta la fel.

Pentru a rezolva această problemă fără a schimba felul în care funcţionează programul de navigare, siturile cu muzică au venit cu următoarea schemă. Fişierul legat la titlul cântecului nu este fişierul real cu muzică. În schimb, este ceea ce se numeşte un **metafisier**, adică un fişier foarte scurt cu numele melodiei. Un metafişier tipic poate avea doar o singură linie de text ASCII şi arată astfel:

```
rtsp://joes-audio-server/song-0025.mp3
```

Când programul de navigare primeşte fişierul de o linie, îl scrie pe disc într-un fişier auxiliar, porneşte programul pentru redarea fişierelor media ca pe o aplicaţie ajutătoare şi îi trimite acestuia numele fişierului auxiliar, ca de obicei. Programul pentru redarea fişierelor media citeşte fişierul şi vede că acesta conţine un URL. Apoi contactează *joes-audio-server* şi cere cântecul. Observaţi că programul de navigare nu mai face parte din circuit.

În cele mai multe cazuri, serverul numit în metafişier nu este acelaşi cu serverul Web. De fapt, de obicei nu este nici măcar un server HTTP, ci un server specializat pe media. În acest exemplu, serverul media foloseşte **RTSP** (**Real Time Streaming Protocol –** protocolul pentru fluxuri în timp real), indicat de numele *rtsp* al schemei. Acesta este descris în RFC 2326.

Programul de redare a fişierelor media are patru lucruri importante de făcut:

1. Controlează interfaţa cu utilizatorul.
2. Tratează erorile de transmisie.
3. Decomprimă melodia.
4. Elimină fluctuaţiile.

Majoritatea programelor de redare a fişierelor media din zilele noastre au o interfaţă captivantă cu utilizatorul, uneori simulând o unitate stereo, cu butoane, mânere, glisoare şi afişaje vizuale. Adesea există panouri frontale interschimbabile, numite **învelitori**, pe care utilizatorul le poate suprapune peste panoul programului de redare. Programul de redare trebuie să controleze toate acestea şi să interacţioneze cu utilizatorul.

A doua funcţie a sa este tratarea erorilor. Transmisia de muzică în timp real foloseşte rareori TCP deoarece o eroare şi o retransmisie pot introduce o pauză inacceptabil de lungă în melodie. În locul acestuia, transmisia reală se face de obicei cu un protocol asemănător cu RTP, pe care l-am studiat în Cap. 6. Ca majoritatea protocoalelor de timp real, RTP utilizează UDP şi deci se pot pierde pachete. Programul de redare este cel care trebuie să trateze acest lucru.

În unele cazuri, muzica este întreţesută pentru a face tratarea erorilor mai uşoară. Spre exemplu, un pachet poate conţine 220 de eşantioane stereo, fiecare conţinând o pereche de numere de 16 biţi, de obicei bune pentru 5 ms de muzică. Dar protocolul poate trimite toate eşantioanele impare pentru un interval de 10 ms într-un pachet şi toate eşantioanele pare în următorul. Atunci un pachet pierdut nu reprezintă o pauză de 5 ms în muzică, ci pierderea tuturor celorlalte eşantioane pentru 10 ms. Această pierdere poate fi tratată uşor de programul de redare a fişierelor media interpolând eşantioanele anterioare şi următoare pentru a estima valoarea absentă.

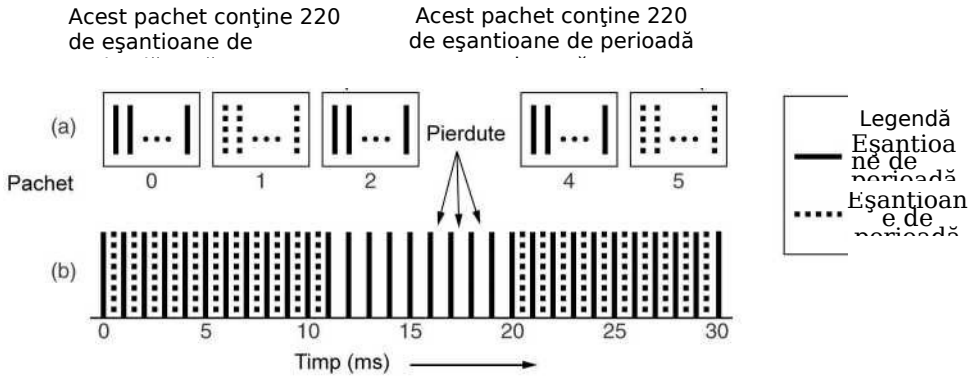


Fig. 7-60. Când pachetele conţin eşantioane alternante, pierderea unui pachet reduce temporar rezoluţia în loc să creeze o perioadă de pauză.

Folosirea întreţeserii pentru a recupera din erori este ilustrată în fig. 7-60. Aici fiecare pachet reţine eşantioanele alternante în timp pentru un interval de 10 ms. În consecinţă, pierderea pachetului 3, după cum este arătat, nu introduce o pauză în muzică, ci doar scade rezoluţia pentru un anume interval. Valorile absente pot fi interpolate pentru a oferi sunet continuu. Această schemă particulară nu funcţionează decât cu eşantioane necomprimate, dar arată cum o codificare inteligentă poate converti un pachet pierdut într-unul de calitate mai joasă, mai degrabă decât într-o pauză de timp. În orice caz, RFC 3119 dă o schemă care funcţionează cu date audio comprimate.

A treia funcţie a programului de redare a fişierelor media este decomprimarea melodiei. Deşi această operaţie necesită multe resurse, ea este relativ rapidă.

A patra funcţie este eliminarea fluctuaţiilor, blestemul tuturor sistemelor în timp real. Toate sistemele de fluxuri audio încep prin a depune într-un tampon 10-15 sec. de muzică înainte de a începe să cânte, ca în fig. 7-61. Ideal, serverul va continua să umple tamponul cu aceeaşi viteză cu care este golit de programul de redare a fişierelor media, însă în realitate nu se întâmplă aşa, astfel că se poate recurge la umplerea tamponului în interiorul buclei.

Două soluţii pot fi adoptate pentru a ţine tamponul plin. Cu un **server de cereri** (pull server), atâta timp cât este loc în tampon pentru încă un bloc, programul de redare continuă să trimită către server cereri pentru câte un bloc suplimentar. Scopul său este să ţină tamponul cât mai plin posibil.

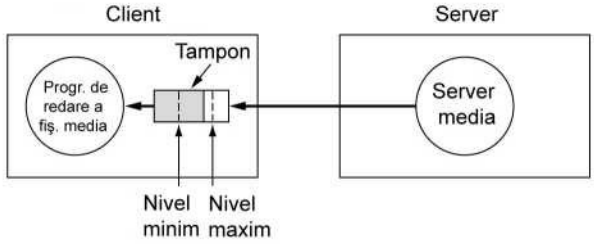


Fig. 7-61. Programul de redare a fişierelor media stochează intrarea de la serverul media şi reproduce din tampon în loc să reproducă direct de pe reţea.

Dezavantajul unui server de cereri este existenţa unor cereri de date inutile. Serverul ştie că a trimis tot fişierul, aşa că de ce să punem programul de redare să întrebe încontinuu? Din acest motiv, soluţia este folosită rar.

Cu un **server de forţare** (push server), programul de redare a fişierelor media trimite o cerere *PLAY,* iar serverul nu face decât să îi trimită date încontinuu. Aici sunt două posibilităţi: serverul media rulează cu viteză normală de playback sau rulează mai repede. În ambele cazuri, unele date sunt puse în tampon înainte de a începe playback-ul. Dacă serverul rulează la viteza normală de playback, datele care vin de la acesta sunt adăugate la sfârşitul tamponului, iar programul de redare şterge datele de la începutul tamponului. Atâta timp cât totul funcţionează perfect, cantitatea de date din tampon rămâne constantă în timp. Această schemă este simplă, deoarece nu sunt necesare mesaje de control în nici o direcţie.

Cealaltă schemă de forţare este cea în care serverul trimite date mai repede decât este nevoie. Avantajul acesteia este că dacă nu se poate garanta că serverul rulează cu o viteza constantă, acesta are ocazia să recupereze de fiecare dată când rămâne în urmă. O problemă este posibila depăşire a cantităţii tamponului dacă serverul trimite date mai repede decât sunt consumate (şi trebuie să poată face asta pentru a putea evita pauzele).

Soluţia este ca programul de redare a fişierelor media să definească un **nivel minim** (low-water mark) şi un **nivel maxim** (high-water mark) în tampon. Practic, serverul trimite date până când tamponul este umplut la nivelul maxim. Apoi programul de redare a fişierelor media îi spune să ia o pauză. Cum datele vor continua să intre în tampon până când serverul primeşte cererea de pauză, distanţa de la nivelul maxim până la sfârşitul tamponului trebuie să fie mai mare decât întârzierea produsă de lăţimea de bandă a reţelei. După ce serverul este oprit, tamponul începe să se golească. Când ajunge la nivelul minim, programul de redare a fişierelor media îi spune serverului media să reia trimiterea. Nivelul minim trebuie poziţionat astfel încât tamponul să nu se golească integral.

Pentru a acţiona asupra serverului de forţare, programul de redare a fişierelor media trebuie să îl controleze de la distanţă. Acest lucru este asigurat de RTSP. Acesta este definit în RFC 2326 şi oferă mecanisme de control al serverului pentru programul de redare. El nu se ocupă de fluxul de date, lucru făcut de obicei de RTP. Comenzile principale oferite de RTSP sunt listate în fig. 7-62.

Comanda	Răspunsul serverului
DESCRIBE	Afişează parametrii media
SETUP	Stabileşte un canal logic între programul de redare şi
PLAY	Începe să trimită date clientului.
RECORD	Începe să accepte date de la client.
PAUSE	Suspendă temporar trimiterea datelor.
TEARDOWN	Eliberează canalul logic.

Fig. 7-62. Comenzile RTSP de la programul de redare la server.

7.4.4 Radio prin Internet

O dată ce a devenit posibilă transmiterea de fluxuri audio prin Internet, staţiilor radio comerciale le-a venit ideea să emită şi pe Internet, şi prin aer. Nu cu mult timp după asta, staţiile radio ale universităţilor au început să îşi pună semnalul pe Internet. Apoi *studenţii* şi-au înfiinţat propriile staţii radio. Cu tehnologia actuală, practic oricine îşi poate înfiinţa o staţie radio. Întreaga zonă a radioului prin Internet este foarte nouă şi în schimbare, însă merită să spunem câte ceva despre ea.

Există două soluţii generale pentru radio-ul prin Internet. În prima, programele sunt pre-înregis- trate şi stocate pe disc. Ascultătorii se pot conecta la arhivele staţiei radio şi pot alege şi descărca orice program, pentru a-l asculta. De fapt, aceasta este similară cu fluxurile audio despre care tocmai am discutat. Este de asemenea posibil să se stocheze fiecare program imediat după ce a fost transmis în direct, astfel încât arhiva să funcţioneze doar pentru, să zicem, o jumătate de oră sau mai puţin după transmisia în direct. Avantajele acestei soluţii sunt că este uşor de realizat, toate tehnicile despre care am discutat funcţionează şi aici, iar ascultătorii pot alege dintre toate programele din arhivă.

Cealaltă soluţie este transmiterea în direct pe Internet. Unele staţii transmit prin aer şi prin Internet simultan, dar sunt din ce în ce mai multe staţii radio exclusiv pe Internet. Unele tehnici care sunt aplicabile fluxurilor audio sunt aplicabile şi radio-ului în direct prin Internet, dar sunt şi unele diferenţe cheie.

Un element asemănător este necesitatea stocării într-un tampon în situl utilizatorului pentru a micşora fluctuaţiile. Colectând 10 sau 15 secunde de radio înainte de începerea playback-ului, sunetul poate fi menţinut continuu chiar şi în cazul unor fluctuaţii substanţiale pe reţea. Atât timp cât pachetele ajung înainte să fie nevoie de ele, nu contează când au ajuns.

O diferenţă de bază este că fluxurile audio pot fi transmise cu o viteză mai mare decât viteza de playback, întrucât receptorul le poate opri când este atins nivelul maxim. Teoretic, asta îi dă timp pentru a retransmite pachetele pierdute, deşi această abordare nu se foloseşte de obicei. În contrast, radio-ul în direct este întotdeauna transmis la exact aceeaşi rată cu care este generat şi redat ascultătorului.

O altă diferenţă este că o staţie radio în direct are de obicei sute sau mii de ascultători simultan, în timp ce fluxurile audio sunt punct la punct. În aceste condiţii, radio-ul prin Internet ar trebui să folosească trimiterea multiplă cu

protocoalele RTP/RTSP. Aceasta este cu siguranţă cea mai eficientă cale de a funcţiona.

În practica actuală, radio-ul prin Internet nu funcţionează aşa. Ceea ce se întâmplă de fapt este că utilizatorul stabileşte o conexiune TCP cu staţia şi fluxul este trimis prin conexiunea TCP. Fireşte, asta creează diverse probleme, precum oprirea fluxului când fereastra este plină, pierderea pachetelor care au expirat şi sunt retransmise, şi aşa mai departe.

Motivul pentru care se foloseşte trimiterea singulară TCP în locul trimiterii multiple RTP are trei părţi. Prima, puţine ISP-uri suportă trimiterea multiplă, astfel că aceasta nu este practic o opţiune. A doua, RTP este mai puţin cunoscut decât TCP şi staţiile radio sunt adesea mici şi au puţine cunoştinţe despre calculatoare, aşa încât este mai uşor de folosit un protocol înţeles pe scară largă şi suportat de toate pachetele de aplicaţii. A treia, mulţi oameni ascultă radio-ul prin Internet la serviciu, ceea ce, în practică, înseamnă adesea în spatele unui zid de protecţie (firewall). Mulţi administratori de sistem îşi configurează zidul de protecţie pentru a-şi proteja LAN-ul de vizitatori nepoftiţi. De obicei ei acceptă conexiuni TCP de la portul de la distanţă 25 (SMTP pentru poştă electronică), pachete UDP de la portul de la distantă 53 (DNS), şi conexiuni TCP de la portul de la distanţă 80 (H'ITP pentru Web). Aproape orice altceva poate fi blocat, inclusiv RTP. Astfel, singura cale de a obţine semnalul radio prin zidul de protecţie este ca situl Web să pretindă că este un server HTTP, cel puţin în faţa zidului de protecţie, şi să folosească servere HTTP care utilizează TCP. Aceste măsuri severe, în timp ce oferă doar o securitate minimală, împing adesea cu forţa aplicaţiile multimedia către moduri de operare cu o eficienţă drastic mai mică.

Cum radio-ul prin Internet este un mediu nou, războaiele asupra formatelor sunt în plină înflorire. Real Audio, Windows Media Audio şi MP3 concurează agresiv pe această piaţă pentru a deveni formatul dominant în radio-ul prin Internet. Un nou venit este Vorbis, care este tehnic similar cu MP3, însă are sursele disponibile şi este suficient de diferit pentru a nu folosi patentele pe care se bazează MP3.

O staţie tipică de radio prin Internet are o pagină Web ce îi listează programul, informaţii despre DJ şi crainicii săi şi multe reclame. Sunt de asemenea şi una sau mai multe iconiţe pentru a afişa formatele audio pe care le suportă (sau doar ASCULTĂ ACUM dacă un singur format este suportat). Aceste iconiţe sau ASCULTĂ ACUM sunt metafişiere legate de tipul celor de care am discutat mai sus.

Când un utilizator selectează una dintre iconiţe cu mouse-ul, este trimis metafişierul. Programul de navigare îi foloseşte tipul MIME sau extensia de fişier pentru a determina aplicaţia ajutătoare cea mai potrivită pentru metafişier (spre exemplu programul de redare a fişierelor media). Apoi scrie metafişierul într-un fişier auxiliar pe disc, porneşte programul de redare a fişierelor media şi îi trimite numele fişierului auxiliar. Programul de redare a fişierelor media citeşte fişierul auxiliar, vede URL-ul pe care acesta îl conţine (de obicei o schemă *http* în loc de *rtsp* pentru a evita problema zidului de protecţie şi pentru că unele aplicaţii multimedia de succes lucrează astfel), contactează serverul şi începe să funcţioneze ca un radio. Ca element exterior, audio conţine un singur flux, astfel că *http*-ul funcţionează, însă pentru video, care are cel puţin două fluxuri, *http*-ul nu mai e

bun şi este necesar ceva de genul *rtsp.*

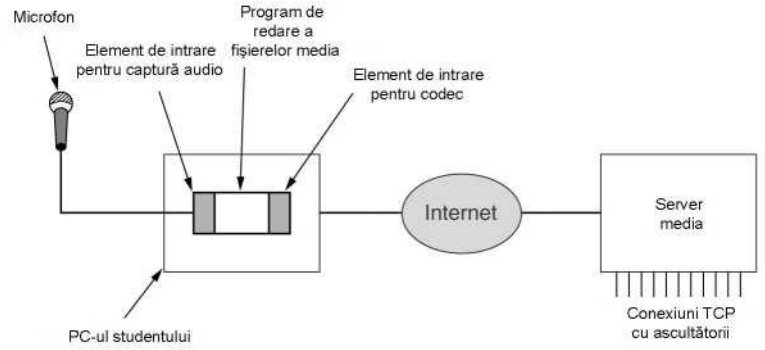


Fig. 7-63. O staţie radio student.

O altă dezvoltare interesantă în zona radio-ului prin Internet este o organizare prin care oricine, chiar şi un student, poate să înfiinţeze şi să opereze o staţie radio. Componentele principale sunt ilustrate în fig. 7-63. Elementul de bază al staţiei este un calculator normal cu o placă de sunet şi un microfon. Aplicaţiile constau într-un program de redare a fişierelor media, precum Winamp sau Freeamp, cu un element de intrare pentru captură audio şi un codec pentru formatul audio de ieşire selectat, spre exemplu MP3 sau Vorbis.

Fluxul audio generat de staţie este apoi trimis prin Internet către un server mare, care se ocupă de distribuţia acestuia unui număr mare de conexiuni TCP. De obicei serverul suportă multe staţii mici. Acesta menţine de asemenea un director cu staţiile pe care le are şi ce emite fiecare în momentul curent. Potenţialii ascultători merg la server, selectează o staţie, şi primesc date TCP. Există pachete de aplicaţii comerciale pentru controlul tuturor parţilor, precum şi pachete cu sursele disponibile precum icecast. Există de asemenea şi servere care sunt doritoare să se ocupe de distribuţie contra unei taxe.

7.4.5 Voce peste IP

Cu mult timp în urmă, sistemul telefonic public comutat era în primul rând utilizat pentru traficul de voce cu variaţii mici de trafic de date. Dar traficul de date a crescut din ce în ce mai mult, încât în 1999, numărul de biţi de date transportaţi era egal cu numărul de biţi de voce (deoarece vocea este în PCM pe legăturile principale, poate fi măsurată în biţi/sec). Până în 2002, volumul de trafic de date era cu un ordin de mărime mai mare decât volumul de trafic de voce şi încă creştea exponenţial, traficul de voce fiind aproape la acelaşi nivel (5% creştere pe an).

Ca o consecinţă a acestor numere, mulţi operatori de reţele de pachete comutate au devenit dintr-o dată interesaţi de transportul vocii prin reţelele lor de date. Cantitatea suplimentară de bandă de transfer necesară pentru voce este minusculă, din moment ce reţelele de pachete au o dimensiune specifică traficului de date. Cu toate acestea, nota de plată a unei persoane este mai mare pentru telefon, decât pentru Internet, în felul acesta operatorii de reţea văzând telefonia

pe Internet ca un mod de a câştiga mai mulţi bani fără să mai pună alte cabluri în pământ. Astfel a luat naştere **telefonia pe Internet** (cunoscută şi sub numele de **voce peste IP**).

H.323

Un lucru era clar pentru toată lumea încă de la început şi anume că dacă fiecare vânzător şi-ar fi creat propria stivă de protocoale, sistemul nu ar fi funcţionat niciodată. Pentru a evita această problemă, un număr de participanţi interesaţi s-au adunat sub auspiciile ITU pentru a realiza standardele. În 1996 ITU a lansat o recomandare **H.323** intitulată "Sisteme de telefonie vizuală şi echipamente pentru reţele locale care nu garantează calitatea serviciului". Doar industria telefonică ar putea gândi un astfel de nume. Recomandarea a fost revizuită în 1998 şi apoi acest H.323 revizuit a fost baza primelor sisteme universale de telefonie pe Internet.

H.323 este mai mult o prezentare arhitecturală a telefoniei pe Internet decât un protocol specific. El se referă la un număr mare de protocoale specifice pentru codificarea vocii, configurarea apelului, semnalizare, transportul datelor şi alte aspecte mai mult decât să specifice el însuşi aceste lucruri. Modelul general este reprezentat în fig. 7-64. În centru este o **poartă (gateway)** care conectează la Internet reţeaua de telefonie. Comunică prin protocoalele H.323 pe partea de Internet şi prin protocoalele PSTN pe partea de telefonie. Dispozitivele de comunicaţie sunt denumite **terminale** . Un LAN poate avea un **administrator de poartă (gatekeeper)**, care controlează punctele finale de sub jurisdicţia sa, numite **zone.**

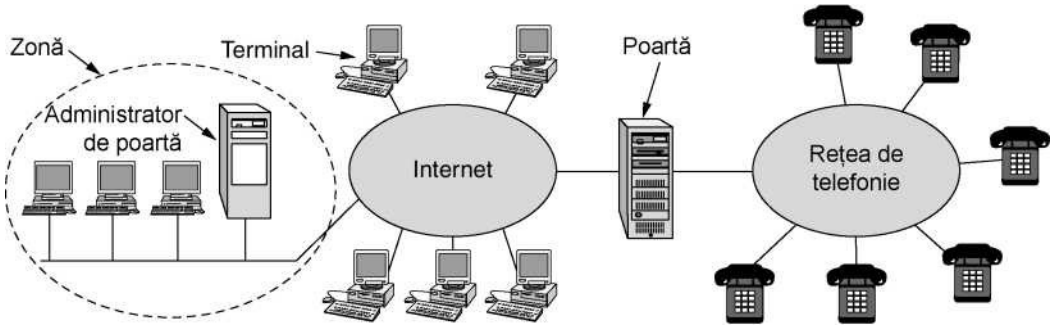


Fig. 7-64. Modelul arhitectural H.323 pentru telefonia prin Internet.

O reţea de telefonie are nevoie de un număr de protocoale. Să începem cu protocolul de codificare şi decodificare a vocii. Sistemul PCM pe care l-am studiat în Cap. 2 este definit în recomandarea ITU **G.711**. Acesta codifică un singur canal de voce prin eşantionări de 8000 de ori pe secundă cu un model pe 8 biţi care să redea vocea necomprimată la 64 Kbps. Toate sistemele H.323 trebuie să suporte G.711. Cu toate acestea, alte protocoale de compresie a vocii sunt de asemenea permise (dar nu obligatorii). Acestea folosesc diferiţi algoritmi de compresie pentru a face diferite compromisuri între calitate şi lărgime de bandă. De exemplu, **G.723.1** preia un bloc de 240 de eşantioane (30 ms de voce) şi foloseşte codificarea predictivă pentru a-l reduce la 24 octeţi sau 20 octeţi. Acest algoritm oferă o rată la ieşire de 6,4 Kbps sau 5,3 Kbps (factori de compresie de 10 şi 12), respectiv, cu mici pierderi în calitatea percepută. Sunt, de asemenea, permise şi alte codificări.

Din moment ce sunt permişi mai mulţi algoritmi de compresie, este necesar un protocol care să permită terminalelor să negocieze ce algoritm vor folosi. Acest protocol poartă numele de **H.245**. De asemenea, se negociază şi alte aspecte ale conexiunii, ca de exemplu viteza de transmisie. RTCP este necesar pentru controlul canalelor RTP. De asemenea, este necesar un protocol pentru stabilirea şi eliberarea conexiunilor, asigurarea de tonuri, crearea de sunete de apel şi restul telefoniei standard. Aici este utilizat ITU **Q.931**. Terminalele au nevoie de un protocol pentru a comunica cu administratorul de poartă (când acesta există). În acest scop este folosit **H.255**. Canalul PC – administrator de poartă pe care îl gestionează este denumit canal **RAS (Registration/Admission/Status, rom: Înregistrare/Admisie/Stare)**. Acest canal permite terminalelor să intre sau să părăsească zona, să ceară sau să elibereze bandă de transfer şi să asigure, printre altele, actualizări de stare. În fine, un protocol este necesar pentru transmiterea efectivă a datelor. RTP este utilizat în acest scop. Este administrat de RTCP, ca de obicei. Poziţionarea tuturor protocoalelor este prezentată în fig. 7-65.

<table>
<tr><td>Voce</td><td colspan="4">Control</td></tr>
<tr><td>G.7xx
RTP</td><td>RTCP</td><td>H.225 (RAS)</td><td>Q.931 (semnalizare apel)</td><td>H.245 (control apel)</td></tr>
<tr><td colspan="3">UDP</td><td colspan="2">TCP</td></tr>
<tr><td colspan="5">IP</td></tr>
<tr><td colspan="5">Protocolul de legătură de date</td></tr>
<tr><td colspan="5">Protocolul de nivel fizic</td></tr>
</table>

Fig. 7-65. Stiva de protocoale H.323.

Pentru a vedea cum lucrează împreună aceste protocoale, să considerăm cazul unui terminal PC pe un LAN (cu un administrator de poartă) care apelează un telefon aflat la distanţă. Mai întâi, PC- ul trebuie să localizeze administratorul de poartă, astfel că difuzează un pachet UDP de aflare a administratorului de poartă pe portul 1718. Când administratorul de poartă răspunde, PC-ul află adresa IP a administratorului de poartă. Acum PC-ul se înregistrează la administratorul de poartă trimiţându-i un mesaj RAS într-un pachet UDP. După ce a fost acceptat, PC-ul trimite administratorului de poartă un mesaj de admitere RAS, cerând lărgime de bandă. Numai după ce banda a fost acordată se poate face iniţierea apelului. Ideea de a cere lărgime de bandă în avans este aceea de a permite administratorului de poartă să limiteze numărul de apeluri pentru a evita supraîncărcarea liniei de ieşire şi a ajuta la asigurarea calităţii necesare a serviciului.

În acest moment, PC-ul stabileşte o conexiune TCP cu administratorul de poartă pentru a iniţia apelul. Configurarea apelului foloseşte protocoale existente de reţea de telefonie, care sunt orientate pe conexiuni, deci este necesar TCP-ul. În contrast, sistemul telefonic nu are nimic echivalent canalului RAS pentru a permite telefoanelor să-şi anunţe prezenţa, astfel creatorii H.323 au fost nevoiţi să folosească fie UDP, fie TCP pentru RAS, şi au ales protocolul cu supraîncărcarea cea mai mică, UDP.

În acest moment, când PC-ul are alocată bandă de transfer, el poate să trimită

un mesaj Q.931 *SETUP* (configurare) peste conexiunea TCP. Acest mesaj specifică numărul de telefon apelat (sau adresa IP şi portul, dacă este apelat un calculator). Administratorul de poartă răspunde cu un mesaj Q.931 *CALL PROCEEDING* (începerea comunicării) pentru a confirma primirea cererii. Administratorul de poartă trimite mai departe mesajul *SETUP* către poartă.

Poarta, care este jumătate calculator, jumătate comutator telefonic, lansează un apel obişnuit către telefonul dorit. Oficiul final la care este legat telefonul, apelează telefonul destinaţie şi trimite de asemenea un mesaj Q.931 *ALERT* (alertă) pentru a anunţa PC-ul apelant că a început să sune. Când persoana de la celălalt capăt al firului ridică receptorul, oficiul final trimite înapoi un mesaj Q.931 *CONNECT* (conectare) pentru a anunţa PC-ul că are o conexiune.

Odată stabilită conexiunea, administratorul de poartă nu mai este în buclă, deşi poarta încă mai este. Pachetele următoare trec peste administratorul de poartă, ducându-se direct la adresa IP a porţii. În acest punct, avem doar un simplu tub între cele două părţi. Aceasta este doar o conexiune la nivel fizic pentru transferul biţilor, nimic mai mult. Nici un capăt nu ştie nimic despre celălalt.

Protocolul H.245 este acum folosit pentru negocierea parametrilor apelului. El foloseşte canalul de control H.245, care este întotdeauna deschis. Fiecare parte începe prin anunţarea capabilităţilor sale, de exemplu, dacă poate suporta apeluri video (H.323 suportă apeluri video) sau conferinţe, ce codificări suportă etc. În momentul în care fiecare capăt ştie ce suportă celălalt, sunt stabilite două canale unidirecţionale şi un codor şi alţi parametri sunt atribuiţi fiecăruia. Din moment ce fiecare capăt poate avea un echipament diferit, este foarte probabil să fie diferite şi codoarele pe canalele de trimitere şi recepţie. După ce s-au încheiat toate negocierile, fluxul de date utilizând RTP poate începe. El este administrat prin RTCP, care joacă un rol în controlul congestiei. Dacă sunt prezente transmisii video, RTCP se ocupă de sincronizarea audio/video. Diferitele canale sunt ilustrate în fig. 7-66. Când oricare din cele două capete se închide, canalul Q.931 de semnalizare al apelului este utilizat pentru a opri conexiunea.

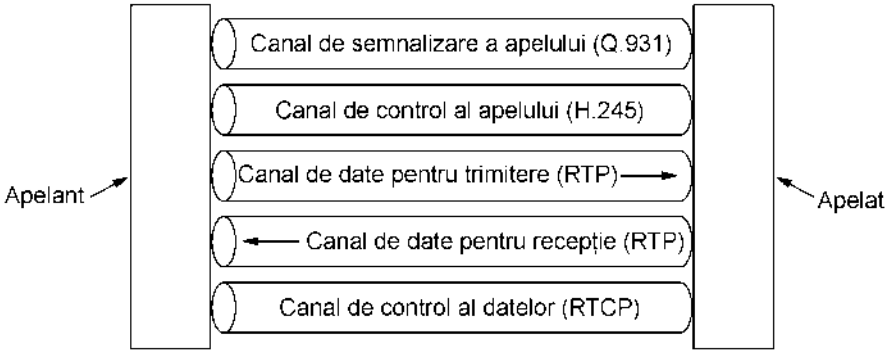

Canal de semnalizare a apelului (Q.931)
Canal de control al apelului (H.245)
Canal de date pentru trimitere (RTP)
Canal de date pentru recepție (RTP)
Canal de control al datelor (RTCP)
Apelant
Apelat

Fig. 7-66. Canale logice între apelant şi apelat în timpul apelului.

Când se încheie apelul, PC-ul apelant contactează din nou administratorul de poartă cu un mesaj RAS pentru a elibera banda de transfer care i-a fost atribuită . Alternativ, el poate să facă un nou apel.

Nu am spus nimic despre calitatea serviciului, deşi aceasta este esenţială pentru a face din vocea peste IP un succes. Motivul este simplu, calitatea serviciului este în afara domeniului H.323. Dacă reţeaua transportatoare este capabilă să producă o conexiune stabilă, fără distorsiuni dinspre PC-ul apelant (de exemplu, folosind tehnicile pe care le-am discutat în cap. 5) înspre poartă, atunci calitatea serviciului pe apel va fi bună; altfel, nu va fi. Partea telefonică foloseşte PCM şi este întotdeauna fără distorsiuni.

SIP – Protocolul de iniţiere a sesiunii

H.323 a fost conceput de ITU. Mulţi oameni din comunitatea Internet l-au văzut ca un produs tipic telco: mare, complex, şi inflexibil. În consecinţă, IETF a creat un comitet pentru a concepe un mod mai simplu şi mai modular pentru vocea peste IP. Rezultatele cele mai bune până în prezent se concretizează în **SIP (Session Initiation Protocol, rom: Protocolul de iniţiere a sesiunii)**, care este descris în RFC 3261. Protocolul descrie configurarea apelurilor telefonice pe Internet, video conferinţele şi alte conexiuni multimedia. Spre deosebire de H.323, care este o întreagă suită de protocoale, SIP este un singur modul, dar a fost conceput pentru a conlucra bine cu aplicaţiile Internet existente. De exemplu, defineşte numerele de telefon ca URL-uri, pentru a fi incluse în pagini Web, permiţând ca un clic pe o legătură să iniţieze un apel telefonic (asemănător, schema *mailto* permite ca activarea unei hiper-legături să deschidă programul de trimitere al unui mesaj electronic).

SIP poate stabili sesiuni bilaterale (apeluri telefonice obişnuite), sesiuni multilaterale (în care oricine poate auzi şi vorbi), şi sesiuni cu transmisie multiplă (un emiţător, mai mulţi receptori). Sesiunile pot conţine audio, video, sau date, ultimul fiind folositor de exemplu pentru jocuri cu mai mulţi utilizatori în timp real. SIP se ocupă doar cu configurarea, administrarea şi terminarea sesiunilor. Alte protocoale, ca RTP/RTCP, sunt utilizate pentru transportul datelor. SIP este un protocol de nivel aplicaţie şi poate rula peste UDP sau TCP.

SIP suportă o varietate de servicii, inclusiv localizarea apelatului (care poate nu este la calculatorul său de acasă) şi să determine capabilităţile acestuia, precum şi să trateze mecanismele de configurare şi terminare. În cel mai simplu caz, SIP setează o sesiune de la calculatorul apelantului la calculatorul apelatului, deci să tratăm mai întâi acest caz.

Numerele de telefon în SIP sunt reprezentate ca URL-uri utilizând schema *sip*, de exemplu, *sip:ilse@cs.university.edu* pentru utilizatorul ilse de pe calculatorul specificat de numele de DNS *cs.university.edu.* URL-urile SIP pot conţine adrese IPv4, IPv6, sau chiar numere de telefon.

Protocolul SIP este un protocol bazat pe text modelat în HTTP. Un capăt trimite un mesaj în text ASCII ce conţine un nume de metodă pe prima linie, urmată de linii adiţionale ce conţin antete pentru transmiterea parametrilor. Multe dintre antete sunt luate din MIME pentru a permite protocolului SIP să conlucreze cu aplicaţiile Internet existente. Cele şase metode definite de specificaţia de bază sunt enumerate în fig. 7-67.

Metoda	Descriere

INVITE	Cerere de iniţiere a unei sesiuni
ACK	Confirmare că o sesiune a fost iniţiată
BYE	Cerere de terminare a unei sesiuni
OPTIONS	Interogarea unui calculator despre capabilităţile sale
CANCEL	Anularea unei cereri în aşteptare
REGISTER	Informarea unui server de redirecţionare despre locaţia curentă a

Fig. 7-67. Metodele SIP definite în specificaţia de bază.

Pentru stabilirea unei sesiuni, apelantul fie creează o conexiune TCP cu apelatul şi trimite un mesaj *INVITE*, fie trimite mesajul *INVITE* într-un pachet UDP. În ambele cazuri, antetele din a doua şi următoarele linii descriu structura corpului mesajului ce conţin capabilităţile apelantului, tipurile de mediu de transmisie şi formatele. Dacă cel apelat acceptă convorbirea, el răspunde cu un cod de răspuns de tip HTTP (un număr de trei digiţi conform grupurilor din fig. 7-42, 200 pentru acceptare). După linia cu codul de răspuns, apelatul poate de asemenea include informaţii despre capabilităţile sale, tipurile de mediu de transmisie şi formate.

Conexiunea este realizată prin mecanismul înţelegerii în trei paşi astfel că, pentru a încheia protocolul, apelantul răspunde cu un mesaj de confirmare a recepţion ării mesajului 200.

Oricare dintre cele două capete pot cere terminarea sesiunii prin trimiterea unui mesaj conţinând metoda *BYE.* Când celălalt capăt trimite confirmarea primirii acestuia, sesiunea este terminată.

Metoda *OPTIONS* este utilizată pentru a interoga o maşină despre propriile sale capabilităţi. Este în general folosită înainte de iniţierea unei sesiuni pentru a afla dacă acea maşină este capabilă de transmisii de voce peste IP sau ce alt tip de sesiune este urmărit.

Metoda *REGISTER* se referă la abilitatea protocolului SIP de a urmări şi a se conecta la un utilizator care nu este acasă. Acest mesaj este trimis unui server de localizare SIP care ţine o evidenţă a locaţiilor utilizatorilor. Acel server poate fi mai târziu interogat pentru a afla locaţia curentă a utilizatorului. Operaţia de redirecţionare este ilustrată în fig. 7-68. Aici, apelantul trimite un mesaj *INVITE* unui server proxy pentru a ascunde posibila redirecţionare. Proxy-ul caută apoi unde este utilizatorul şi îi trimite un mesaj *INVITE.* Apoi se comportă ca un intermediar pentru mesajele următoare în înţelegerea în trei paşi. Mesajele *LOOKUP* şi *REPLY* nu fac parte din SIP; orice protocol convenabil poate fi utilizat, depinzând de tipul de server de localizare utilizat.

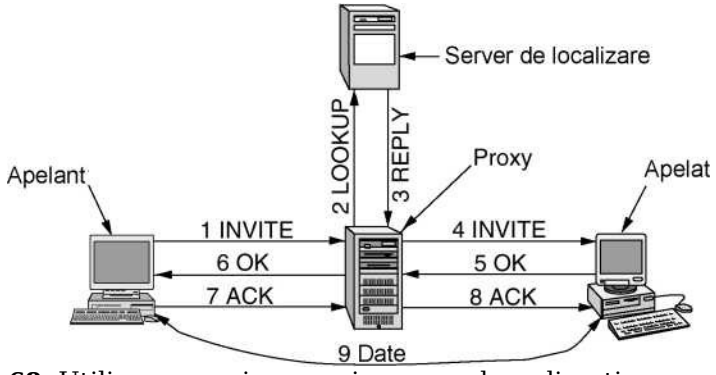


Fig. 7-68. Utilizarea unui proxy şi servere de redirecţionare cu SIP.

SIP are o varietate de alte caracteristici pe care noi nu le vom descrie aici, incluzând aşteptarea apelului, ecranarea apelului, criptare şi autentificare. De asemenea, are abilitatea de a apela de la un calculator, un telefon obişnuit, dacă este disponibilă o poartă de conversie corespunzătoare între Internet şi sistemul telefonic.

Comparaţie între H.323 şi SIP

H.323 şi SIP au multe puncte în comun, dar şi diferenţe. Ambele protocoale permit apeluri bilaterale şi multilaterale folosind atât calculatoare, cât şi telefoane ca puncte finale. Ambele suportă negocierea parametrilor, criptarea şi protocoalele RTP/RTCP. Un rezumat al acestor asemănări şi diferenţe este dat în fig. 7-69.

Noţiune	**H.323**	**SIP**
Conceput de	ITU	IETF
Compatibilitate cu PSTN	Da	În mare parte
Compatibilitate cu Internet	Nu	Da
Arhitectura	Monolitică	Modulară
Completitudine	Întreaga stivă de protocoale	SIP tratează doar configurarea
Negocierea parametrilor	Da	Da
Semnalizarea apelului	Q.931 peste TCP	SIP peste TCP şi UDP
Formatul mesajului	Binar	ASCII
Mediul de transmisie	RTP/RTCP	RTP/RTCP
Apeluri multilaterale	Da	Da
Conferinţe multimedia	Da	Nu
Adresare	Calculator sau număr de	URL
Terminarea apelului	Explicită sau eliberare TCP	Explicită sau la expirarea
Mesagerie imediată	Nu	Da
Criptare	Da	Da
Dimensiunea standardelor	1400 pagini	250 pagini
Implementare	Mare şi complexă	Moderată
Stare	Larg răspândită	Prezentă şi viitoare

Fig. 7-69. Comparaţie între H.323 şi SIP

Deşi seturile de caracteristici sunt similare, cele două protocoale diferă mult în concepţie. H.323 este un standard tipic, cu greutate, al industriei de telefonie, specificând întreaga stivă de protocoale şi definind exact ce este permis şi ce este

interzis. Abordarea duce la protocoale bine definite la fiecare nivel, uşurând interoperabilitatea. Preţul este un standard mare, complex şi rigid, dificil de adaptat la aplicaţiile viitoare.

În contrast, SIP este un protocol tipic de Internet care lucrează prin schimbul de linii scurte de text ASCII. Este un modul uşor care conlucrează bine cu alte protocoale de Internet, dar mai puţin cu protocoalele de semnalizare din sistemul telefonic existent. Deoarece modelul IETF al vocii peste IP este în mare măsură modular, el este flexibil şi poate fi adaptat cu uşurinţă la noi aplicaţii. Partea neplăcută este cea a potenţialelor probleme de interoperabilitate, deşi acestea sunt discutate în întâlniri frecvente unde diverşi implementatori îşi testează împreună sistemele.

Vocea peste IP este o temă prezentă şi viitoare. De aceea, deja există cărţi pe această temă. Câteva exemple sunt (Collins, 2001; Davidson şi Peters, 2000; Kumar ş.a.., 2001; şi Wright, 2001). Ediţia din mai/iunie 2002 a revistei *Internei Computing* are mai multe articole pe această temă.

7.4.6 Introducere la video

Până acum am discutat urechea în detaliu; este timpul să ne mutăm la ochi (nu, această secţiune nu este urmată de una despre nas). Ochiul uman are proprietatea că atunci când o imagine apare pe retină, imaginea este păstrată acolo pentru câteva milisecunde. Dacă o secvenţă de imagini este desenată linie cu linie la 50 imagini/sec, ochiul nu observă că priveşte imagini discrete. Toate sistemele video (de exemplu, televizorul) exploatează acest principiu pentru a produce imagini în mişcare.

Sisteme Analogice

Pentru a înţelege sistemele video, este bine să pornim de la vechea televiziune simplă, alb-negru. Pentru a reprezenta imaginile bidimensionale din faţa ei ca o tensiune unidimensională funcţie de timp, camera de luat vederi scanează imaginea cu o rază electronică, rapid de-a latul şi lent în josul

ei, înregistrând intensitatea luminoasă aşa cum vine. La sfârşitul scanării, numit **cadru** (frame), raza reia traseul. Această intensitate este difuzată ca funcţie de timp, iar receptorii repetă procesul de scanare pentru reconstrucţia imaginii. Modelul de scanare folosit

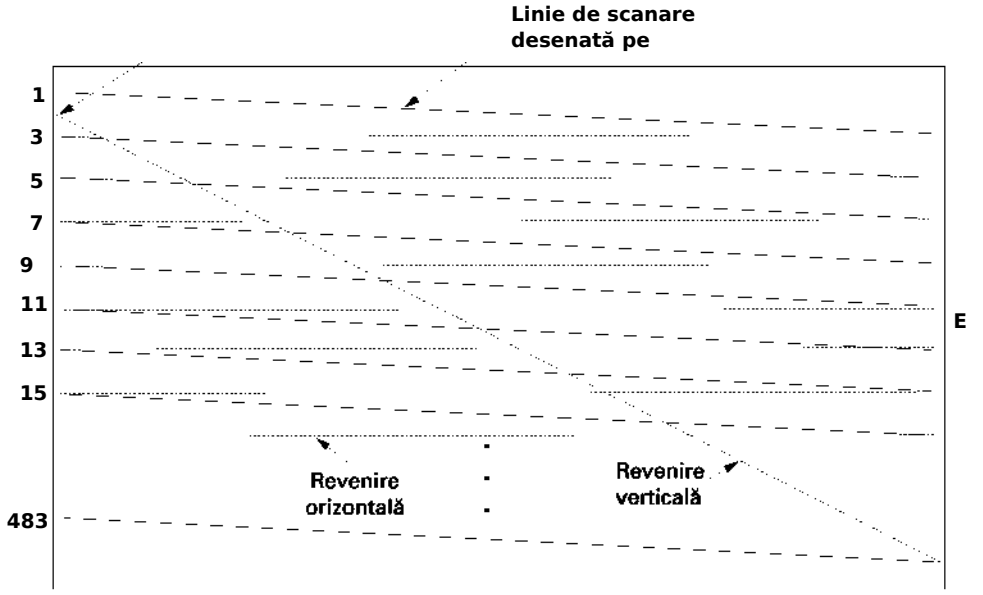


Fig. 7-70. Modelul de scanare folosit pentru video şi

atât de cameră cât şi de receptor, este prezentat în fig. 7-70. (Camerele CCD integrează mai degrabă decât scanează, dar unele camere şi toate monitoarele scanează.)

Următorul câmp
Linie de scanare, începe aici

Parametrii exacţi de scanare variază de la ţară la ţară. Sistemul folosit în America de Nord şi de Sud şi Japonia are 525 de linii de scanare, o rată de aspect orizontal/vertical de 4:3 şi 30 de cadre/sec. Sistemul european are 625 de linii de scanare, aceeaşi rată de aspect de 4:3 şi 25 de cadre/sec. În ambele sisteme, câteva linii din vârf şi câteva din partea de jos nu sunt afişate (pentru a aproxima o imagine dreptunghiulară pe un tub catodic rotund original). Doar 483 din cele 525 de linii de scanare NTSC (şi 576 din cele 625 de linii de scanare PAL/SECAM) sunt afişate. Raza este stinsă în timpul revenirii verticale, aşa că multe staţii (în special în Europa) folosesc acest interval pentru difuzare de TeleText (pagini de text conţinând ştiri, vreme, sporturi, preţuri la bursă etc.).

În timp ce 25 de cadre/sec sunt suficiente pentru a capta o mişcare lină, la această viteză a cadrelor multe persoane, în special cei bătrâni, vor percepe imaginea tremurată (deoarece imaginea veche a fost ştearsă de pe retină înaintea apariţiei uneia noi). În loc să se mărească viteza cadrelor, care va cere să se

folosească mai puţină lărgime de bandă, este aleasă o altă cale. În locul afişării liniilor de scanare în ordine, întâi sunt afişate toate liniile de scanare cu numere impare, apoi cele cu numere pare. Fiecare din aceste jumătăţi de cadre este numită **câmp (field)**. Experimentele arată că deşi oamenii remarcă o pâlpâire la 25 de cadre/sec, ei nu o remarcă la 50 de cadre/sec. Această tehnică este numită **întreţesere (interlacing)**. O televiziune sau video neîntreţesută este denumită **progresivă (progressive)**. Observaţi că filmele rulează la 24 fps, dar fiecare cadru este vizibil în totalitate pentru 1/24 sec.

Video-ul color foloseşte acelaşi model de scanare ca monocromul (alb şi negru), cu excepţia faptului că în locul afişării unei imagini cu o singură rază mişcătoare, sunt folosite trei raze care se mişcă la unison. Este folosită câte o rază pentru fiecare din cele trei culori aditive primare: roşu, verde şi albastru (RGB). Această tehnică funcţionează pentru că orice culoare poate fi construită din super- poziţia liniară a roşului, verdelui şi albastrului cu intensităţile corespunzătoare. Cu toate acestea, pentru transmiterea pe un singur canal, cele trei semnale de culori trebuie combinate într-un singur semnal **compus (composite)**.

Atunci când a apărut televiziunea color, diverse metode de afişare color erau tehnic posibile, şi diverse ţări au făcut alegeri diferite, conducând la sisteme care sunt încă incompatibile. (Observaţi că aceste alegeri nu au nimic în comun cu VHS, faţă de Betamax şi faţă de P2000, care sunt tehnici de înregistrare.) În toate ţările, o necesitate politică a fost ca programele color să fie recepţionate de televizoarele existente alb-negru. În consecinţă, cea mai simplă schemă, care codifică separat semnalele RGB, nu a fost acceptată. De asemenea, RGB nu este cea mai eficientă schemă.

Primul sistem color a fost standardizat în Statele Unite de **Comitetul Naţional de Standarde de Televiziune (National Television Standards Committee)**, care a împrumutat acronimul său standardului: **NTSC**. Televiziunea color a fost introdusă în Europa câţiva ani mai târziu, în momentul în care tehnologia a fost îmbunătăţită substanţial, conducând la sisteme cu mare imunitate la zgomote şi culori mai bune. Acestea sunt numite **SECAM (SEquentiel Couleur Avec Memoire, rom: culoare secvenţială cu memorie),** sistem folosit în Franţa şi în ţările din estul Europei, şi **PAL (Phase Alternating Line, rom: linie cu fază alternată)** folosit în restul Europei. Diferenţa în calitatea culorii între NTSC şi PAL/SECAM a condus la gluma că NTSC înseamnă de fapt că aceeaşi culoare nu apare de două ori (Never Twice the Same Color).

Pentru a permite ca transmisiunile color să fie văzute de receptoarele alb-negru, toate cele trei sisteme combină liniar semnalele RGB într-un semnal de **luminanţă (luminance)**, şi două semnale de **crominanţă (chrominance)**, deşi toate folosesc alţi coeficienţi pentru construcţia acestor semnale din semnale RGB. Interesant, ochiul este mai sensibil la semnalele de luminanţă decât la cele de crominanţă, astfel încât ultimele nu trebuie transmise cu mare acurateţe. Ca rezultat, semnalul luminos poate fi difuzat la aceeaşi frecvenţă ca şi vechiul semnal alb-negru, aşa încât poate fi recepţionat pe televizoarele existente alb-negru. Cele două semnale de crominanţă sunt difuzate în benzi înguste la frecvenţe înalte. Unele televizoare au butoane pentru controlul strălucirii, nuanţei şi saturării (sau strălucirii, tentei şi

culorii) pentru controlul separat al celor trei semnale. Înţelegerea lumi- nanţei şi crominanţei este vitală pentru înţelegerea modului în care funcţionează compresia video.

În ultimii ani, a existat un interes considerabil pentru **HDTV (High Definition TeleVision, rom: televiziunea de înaltă definiţie)**, care produce imagini mai bune dublând numărul liniilor de scanare. Statele Unite, Europa şi Japonia au dezvoltat sisteme HDTV, toate diferite şi toate mutual incompatibile. V-aţi fi aşteptat la altceva? Principiile de bază ale HDTV-ului în termeni de scanare, lumi- nanţă, crominanţă şi altele, sunt similare sistemelor existente. Cu toate acestea, toate cele trei formate au o aceeaşi rată a aspectului de 16:9 în loc de 4:3 pentru a le potrivi mai bine formatului folosit pentru filme (care sunt înregistrate pe 35 mm, cu o rată de aspect de 3:2).

Sisteme digitale

Cea mai simplă reprezentare a video-ului digital este o secvenţă de cadre, fiecare constând dintro grilă dreptunghiulară de elemente de imagine, adică **pixeli**. Fiecare pixel poate fi un singur bit, pentru a reprezenta fie albul, fie negrul. Calitatea unui astfel de sistem este similară cu cea obţinută la transmiterea prin fax a unei fotografii color – adică groaznică. (Încercaţi dacă puteţi; sau fotocopi- aţi o fotografie color la o maşină de copiat care nu rasterizează.)

Următorul pas este de a folosi 8 biţi pe pixel pentru a reprezenta 256 nivele de gri. Această schemă dă o calitate ridicată video-ului alb-negru. Pentru video color, sistemele bune folosesc 8 biţi pentru fiecare din culorile RGB, deşi aproape toate sistemele le amestecă pentru transmitere într-un video compus. În timp ce folosind 24 de biţi per pixel se limitează numărul de culori la 16 milioane, ochiul uman nu poate deosebi atâtea culori, deci nici vorbă de mai multe. Imaginile digitale color sunt produse folosind trei raze de scanare, una pentru fiecare culoare. Geometria este aceeaşi cu cea pentru sistemul analogic din fig. 7-70, exceptând faptul că liniile de scanare continue sunt înlocuite acum de linii formate din pixeli discreţi.

Pentru a produce mişcări line, video-ul digital, la fel ca video-ul analog, trebuie să afişeze cel puţin 25 de cadre/sec. Oricum, deoarece monitoarele de bună calitate ale calculatoarelor rescanează deseori ecranul din imagini memorate de 75 de ori pe secundă sau mai des, întreţeserea nu este necesară şi, în consecinţă, nu este folosită în mod obişnuit. Reafişarea (adică redesenarea) aceluiaşi cadru de trei ori la rând este suficientă pentru eliminarea pâlpâirii.

Cu alte cuvinte, continuitatea unei mişcări este determinată de numărul de imagini *diferite* pe secundă, având în vedere că pâlpâirea este determinată de numărul de ori pe secundă în care este desenat ecranul. Aceşti doi parametri sunt diferiţi. O imagine afişată la 20 de cadre/sec nu va arăta distorsionată, dar va pâlpâi, deoarece un cadru va dispărea de pe retină înainte ca următorul cadru să apară. Un film cu 20 de cadre diferite pe secundă, fiecare dintre acestea fiind desenat de patru ori la rând, nu va pâlpâi, dar va părea distorsionat.

Semnificaţia acestor doi parametri devine mai clară atunci când considerăm lărgimea de bandă pentru transmisia video digitală printr-o reţea. Cele mai multe din monitoarele actuale folosesc rata de aspect de 4:3, astfel încât pot folosi

tuburile ieftine din producţia destinată pieţei televizoarelor. Configuraţiile obişnuite sunt 1024x768, 1280x960, şi 1600x1200. Chiar şi cel mai mic dintre acestea cu 24 biţi pe pixel şi 25 cadre/sec are nevoie să fie alimentat la 472 Mbps. Pentru aceasta ar fi nevoie de un purtător SONET OC-12, şi de altfel rularea unui purtător OC-9 SONET în casa fiecăruia nu este chiar la ordinea zilei. Dublarea acestei rate pentru a evita pâlpâirea este chiar mai puţin de dorit. O soluţie mai bună este transmiterea a 25 cadre/sec pe care calculatorul să le memoreze şi să le afişeze de 2 ori. Televiziunea obişnuită nu foloseşte această strategie, deoarece televizoarele nu au memorie. Şi chiar dacă ar avea memorie, sistemele analogice nu pot fi memorate în RAM fără o conversie prealabilă în formă digitală, care necesită hardware în plus. Ca o consecinţă, întreţeserea este necesară pentru televiziunea obişnuită, dar nu şi pentru video digital.

7.4.7 Compresia video

Ar trebui să fie evident acum că transmisia materialului video în formă necomprimată nu intră în discuţie. Singura speranţă este într-o compresie puternică. Din fericire, numeroase cercetări de câteva decenii au ajuns la mai multe tehnici de compresie şi algoritmi care fac posibilă transmisia de multimedia. În această secţiune vom studia cum este realizată compresia video.

Toate sistemele de compresie necesită doi algoritmi: unul pentru comprimarea datelor la sursă şi altul pentru decomprimarea lor la destinaţie. În literatură, aceşti algoritmi sunt denumiţi algoritmi de **codificare** şi **decodificare**. Vom folosi şi aici această terminologie.

Aceşti algoritmi prezintă câteva asimetrii a căror înţelegere este importantă. Mai întâi, pentru multe aplicaţii, un document multimedia, să zicem un film, va fi codificat o dată (atunci când este memorat pe un server multimedia), dar va fi decodificat de milioane de ori (atunci când este vizualizat de clienţi). Această asimetrie înseamnă că este acceptabil ca algoritmul de codificare să fie lent şi să necesite un hardware suplimentar scump, cu condiţia ca algoritmul de decodificare să fie rapid şi să nu ceară un hardware costisitor. Mai mult, operatorul unui server multimedia ar putea dori să închirieze un supercalculator paralel pentru câteva săptămâni, pentru a-şi codifica întreaga videote- că, dar a cere clienţilor să închirieze un supercalculator pentru 2 ore, pentru a vedea un film nu va avea mare succes. Multe sisteme de compresie fac eforturi pentru ca decodificarea să fie rapidă şi simplă, chiar cu preţul încetinirii şi complicării codificării.

Pe de altă parte, pentru multimedia în timp-real, precum conferinţe video, codificarea lentă este inacceptabilă. Codificarea trebuie făcută din mers, în timp-real. În consecinţă, multimedia de timp- real foloseşte algoritmi sau parametri diferiţi faţă de cei utilizaţi pentru memorarea video-urilor pe disc, deseori cu mult mai puţină compresie.

A doua asimetrie este că procesul de codificare/decodificare nu trebuie să fie inversabil. Adică, atunci când se comprimă un fişier, acesta se transmite şi apoi este decomprimat, utilizatorul aşteptând să-l obţină pe cel original, exact până la ultimul bit. Cu multimedia, această cerinţă nu există. De obicei, este acceptabil ca

un semnal video, după codificare şi decodificare, să fie un pic diferit de original. Atunci când ieşirea decodificată nu este egală exact cu intrarea originală, sistemul se spune că este **cu pierderi** (lossy). Dacă intrarea şi ieşirea sunt identice, sistemul este **fără pierderi** (lossless). Sistemele cu pierderi sunt importante, deoarece acceptarea unui număr mic de informaţii pierdute poate oferi un avantaj imens în termenii de rată de compresie posibilă.

Standardul JPEG

Un film este doar o secvenţă de imagini (plus sunet). Dacă am putea găsi un algoritm bun pentru codificarea unei singure imagini, acest algoritm ar putea fi aplicat succesiv fiecărei imagini, pentru a obţine compresia video. Există algoritmi buni de compresie a imaginii, deci să începem acolo studiul nostru despre compresia video. Standardul **JPEG (Joint Photographic Experts Group, rom: grupul comun al experţilor fotografi)** pentru comprimarea imaginilor cu tonuri continue (de exemplu, fotografii), a fost dezvoltat de experţii în fotografii lucrând sub auspiciile ITU, ISO şi IEC, un alt organism de standarde. Este important pentru multimedia deoarece la o primă aproximare, standardul multimedia pentru filme, MPEG, este codificarea JPEG a fiecărui cadru separat, plus câteva caracteristici pentru comprimarea între cadre şi detectarea mişcării. JPEG este definit în Standardul International 10918.

JPEG are patru moduri şi multe opţiuni. Standardul este mai asemănător cu o listă de cumpărături decât cu un simplu algoritm. Pentru scopurile noastre, doar modul secvenţial cu pierderi este relevant, iar acesta este ilustrat în fig. 7-71. Cu toate acestea, ne vom concentra asupra modului în care JPEG este folosit în mod normal pentru codificarea imaginilor video de 24-biţi RGB şi vom lăsa la o parte detalii minore pentru simplitate.

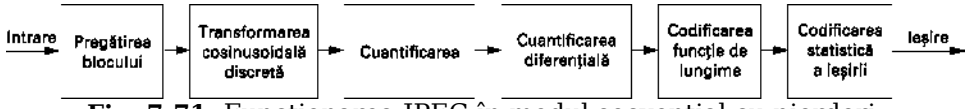


Fig. 7-71. Funcţionarea JPEG în modul secvenţial cu pierderi.

Pasul 1 de codificare a imaginii cu JPEG este pregătirea blocului. Pentru specificare, să presupunem că intrarea JPEG este o imagine RGB de 640x480 cu 24 biţi/pixel, ca în fig. 7-72(a). Deoarece folosirea luminanţei şi crominanţei dă o mai bună compresie, vom calcula mai întâi luminanţa, Y, şi cele două crominanţe, I şi Q (pentru NTSC), după formulele următoare:

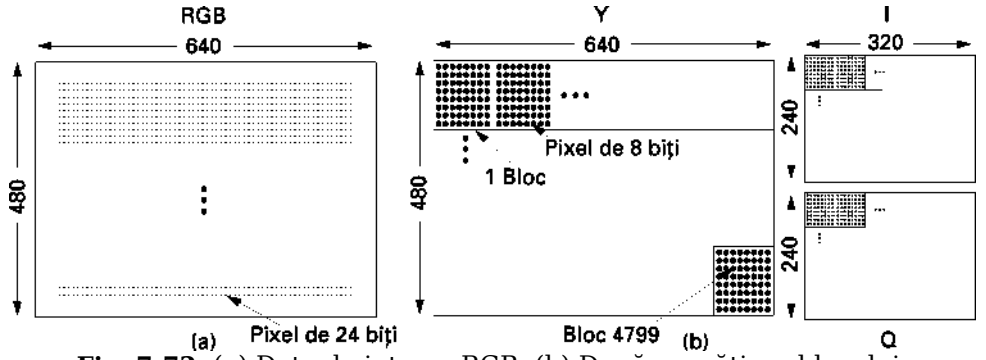


Fig. 7-72. (a) Date de intrare RGB. (b) După pregătirea blocului.

Y = 0,30R + 0,59G + 0,11B
I = 0,60R – 0,28G – 0,32B Q = 0,21R – 0,52G + 0,31B

Pentru PAL, crominanţele sunt notate cu *U* şi *V* şi coeficienţii diferă, dar ideea este aceeaşi. SECAM diferă atât de NTSC cât şi de PAL.

Pentru *YJ* şi *Q* sunt construite matrice separate, fiecare cu elemente între 0 şi 255. Apoi, blocuri pătrate de patru pixeli sunt aranjate în matricele *I* şi *Q* pentru a le reduce la 320x240. Această reducere este cu pierderi, dar ochiul abia dacă observă, deoarece el răspunde la luminanţă mai mult decât la crominanţă. Cu toate acestea, întreaga cantitate de date este comprimată cu un factor de doi. Acum se scade 128 din fiecare element al celor trei matrice pentru a aduce zero la mijlocul intervalului. În fine, fiecare matrice este divizată în blocuri de 8x8. Matricea *Y* are 4800 blocuri; celelalte două au 1200 de blocuri fiecare, aşa cum se arată în fig. 7-72(b).

Pasul 2 al lui JPEG constă în a aplica separat un **DCT (Discrete Cosine Transformation, rom: transformare cosinusoidală discretă)** pentru fiecare din cele 7200 de blocuri. Ieşirea fiecărui DCT este o matrice de 8x8 cu coeficienţii DCT. Elementul DCT (0,0) este valoarea medie a blocului. Celelalte elemente spun câtă putere spectrală este prezentă la fiecare frecvenţă spaţială. Teoretic, DCT este fără pierderi, dar în practică folosirea numerelor în virgulă mobilă şi a funcţiilor transcendente introduce întotdeauna erori de rotunjire care conduc la o mică pierdere de informaţii. În mod normal, aceste elemente se micşorează rapid cu distanţa de la origine, (0, 0), aşa cum este sugerat în fig. 7-73.

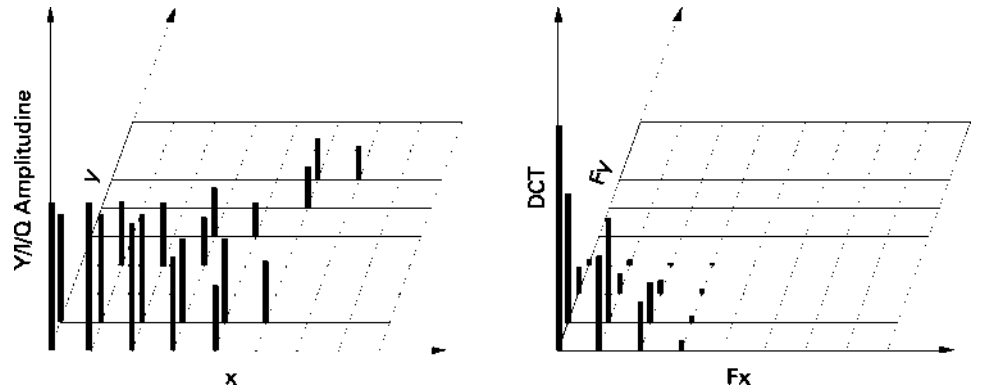


Fig. 7-73. (a) Un bloc al matricei *Y*. (b) Coeficienţii DCT.

Odată ce DCT-ul este complet, JPEG trece la pasul 3, numit **cuantificare (quantization)**, în care coeficienţii DCT cei mai puţin importanţi sunt eliminaţi. Această transformare (cu pierderi) este făcută prin împărţirea fiecărui coeficient din matricea DCT de 8*8 la o pondere luată dintr-o tabelă. Dacă toate ponderile sunt 1, transformarea nu face nimic. Totuşi, atunci când ponderile cresc rapid de la origine, frecvenţele spaţiale înalte sunt eliminate imediat.

Un exemplu al acestui pas este prezentat în fig. 7-74. Aici vedem matricea iniţială DCT, tabela de cuantificare şi rezultatul obţinut prin împărţirea fiecărui element DCT prin elementul corespunzător din tabela de cuantificare. Valorile din tabela de cuantificare nu fac parte din standardul JPEG. Fiecare aplicaţie trebuie să şi le furnizeze, permiţând să se controleze raportul compresie/pierderi.

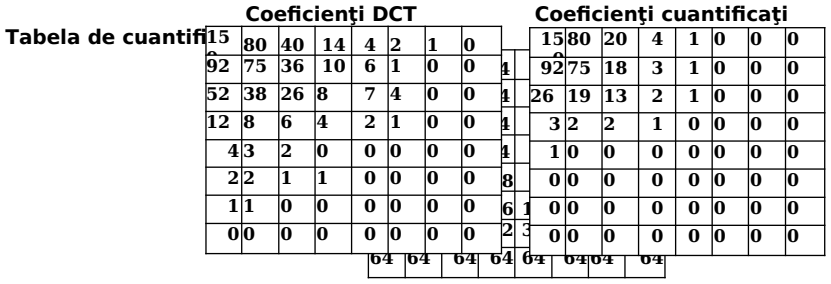


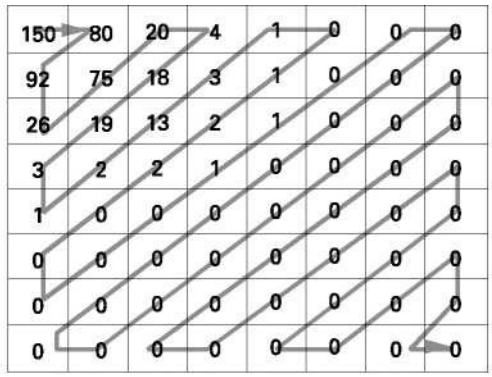
Fig. 7-74. Calculul coeficienţilor DCT cuantificaţi.

Pasul 4 reduce valoarea (0, 0) a fiecărui bloc (cea din colţul stânga sus), înlocuind-o cu diferenţa faţă de elementul corespunzător din blocul precedent. Deoarece aceste elemente sunt mediile respectivelor blocuri, ele trebuie să se modifice lent, astfel încât considerarea valorilor diferenţiale ar trebui să reducă majoritatea dintre ele la valori mici. Diferenţele nu sunt calculate din celelalte valori. Valorile (0,0) sunt numite componente DC; celelalte valori sunt componente AC.

Pasul 5 liniarizează cele 64 de elemente şi aplică listei codificarea după lungimea succesiunilor. Scanarea blocului de la stânga la dreapta şi apoi de sus în jos nu va concentra zerourile împreună, aşa încât se foloseşte un model de căutare în zigzag, ca în fig. 7-75. În acest exemplu, modelul în zigzag produce 38 de 0-uri consecutive la sfârşitul matricei. Acest şir poate fi redus la un singur număr spunând că sunt 38 de 0-uri, tehnică cunoscută sub numele de **codificare după lungimea succesiunilor (run-length encoding)**.

Acum avem o listă de numere care reprezintă imaginea (în spaţiu transformat).

Pasul 6, Huffman, codifică numerele pentru memorare sau transmitere, atribuind numerelor mai des întâlnite coduri mai scurte decât ale celorlalte numere.



7-75. Ordinea de transmitere a valorilor cuantificate

JPEG poate părea complicat, aceasta pentru că el *este* cu adevărat complicat. Chiar şi aşa, deoarece produce o compresie de 20:1 sau mai bună, este larg folosit. Decodificarea unei imagini JPEG cere execuţia algoritmului în sens invers. JPEG este aproape simetric: decodificarea ia acelaşi timp ca şi codificarea. Această proprietate nu este adevărată pentru toţi algoritmii de compresie, aşa cum vom vedea acum.

Standardul MPEG

În fine, ajungem la miezul lucrurilor: standardele **MPEG (Motion Picture Experts Group, rom: grupul experţilor în filme)**. Aceştia sunt algoritmii principali folosiţi pentru compresia video şi sunt standarde internaţionale din 1993. Deoarece filmele conţin atât imagini cât şi sunete, MPEG le poate comprima pe amândouă. Am studiat deja compresia audio şi a imaginilor, deci să examinăm acum compresia video.

Primul standard finalizat a fost MPEG-l (Standard International 11172). Scopul lui a fost de a produce ieşire video de calitatea video recorder-elor (352x240 pentru NTSC) folosind o rată de biţi de 1,2 Mbps. O imagine 352x240 cu 24 biţi/pixel şi 25 cadre/sec are nevoie de 50,7 Mbps, deci reducerea lui la 1,2 Mbps nu este în întregime trivială. MPEG-l poate fi transmis pe linii torsadate la distanţe modeste. MPEG-l este de asemenea folosit pentru memorarea filmelor pe CD-ROM.

Următorul standard din familia MPEG a fost MPEG-2 (Standard International 13818), care a fost proiectat iniţial pentru comprimarea video de calitate de difuzare între 4 şi 6 Mbps, pentru a se potrivi într-un canal de difuzare NTSC sau PAL. Mai târziu, MPEG-2 a fost extins pentru a suporta rezoluţii înalte, incluzând HDTV. Este foarte cunoscut, el stând la baza DVD-ului şi a televiziunii prin satelit.

Principiile de bază ale MPEG-l şi MPEG-2 sunt similare, dar detaliile sunt diferite. La o primă aproximare, MPEG-2 este un superset al lui MPEG-l, cu posibilităţi, formate de cadre, şi opţiuni de codificare suplimentare. Vom discuta mai întâi MPEG-l şi apoi MPEG-2.

MPEG-l are trei părţi: audio, video şi sistem, care le integrează pe celelalte două, ca în fig. 7-76. Codificatoarele audio şi video lucrează independent, ceea ce ridică întrebarea cum se sincronizează cele două fluxuri la receptor. Această problemă se rezolvă având un ceas sistem de 90-kHz, care afişează timpul curent pentru ambele codificatoare. Aceste valori sunt pe 33 de biţi, pentru a permite filmelor să ruleze 24 de ore fără depăşirea valorii maxime. Aceste amprente de timp sunt incluse în ieşirea codificată şi propagate spre receptor, care le poate folosi pentru sincronizare între şirurile video şi audio.

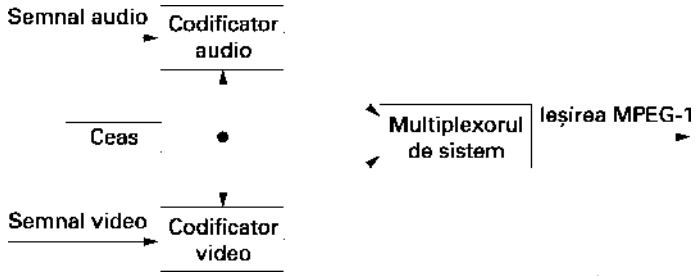


Fig. 7-76. Sincronizarea fluxurilor audio şi video în MPEG-1.

Acum să considerăm compresia video MPEG-1. Există două feluri de redundanţă în filme: spaţială şi temporală. MPEG-l le foloseşte pe amândouă. Redundanţa spaţială poate fi folosită prin simpla codificare separată a fiecărui cadru cu JPEG. Această abordare este câteodată folosită, în special atunci când se folosesc accese aleatorii la acelaşi cadru, ca în editarea producţiilor video. În acest mod, este obţinută o lărgime de bandă comprimată în intervalul 8 Mbps – 10 Mbps.

O comprimare suplimentară poate fi obţinută profitând de faptul că, în general, cadrele consecutive sunt aproape identice. Acest efect este mai mic decât poate apărea la prima vedere, deoarece mulţi fabricanţi de filme taie scenele la fiecare 3 sau 4 secunde (cronometraţi filmul şi număraţi scenele). Cu toate acestea, chiar o serie de 75 de cadre similare oferă potenţialul unei reduceri mari faţă de simpla codificare separată a fiecărui cadru cu JPEG.

Pentru scene în care aparatul de filmat şi fundalul sunt fixe şi unul sau mai mulţi actori se plimbă încet, aproape toţi pixelii vor fi identici de la un cadru la altul. În acest caz, scăzând fiecare cadru din cel precedent şi aplicând JPEG pe diferenţa lor, se obţine un rezultat bun. Cu toate acestea, pentru scenele în care aparatul de filmat micşorează sau măreşte, această tehnică eşuează. Este nevoie de ceva care să compenseze mişcarea. Aceasta este exact ceea ce face MPEG; este principala diferenţă între MPEG şi JPEG.

Ieşirea MPEG-l constă din patru tipuri de cadre:

1. Cadre I (Intracoded) : Fotografii codificate JPEG auto-conţinute.
2. Cadre P (Predictive): Diferenţa bloc cu bloc faţă de ultimul cadru.
3. Cadre B (Bidirectional): Diferenţele faţă de ultimul şi de următorul cadru.
4. Cadre D (DC-coded): Medii ale blocurilor folosite pentru avans rapid.

Cadrele I sunt imagini codificate folosind JPEG, folosind de asemenea luminanţa cu rezoluţie completă şi crominanţa cu jumătate de rezoluţie de-a lungul fiecărei

axe. Este necesar să facem ca aceste cadre I să apară periodic în şirul de ieşire din trei motive. În primul rând, MPEG-l poate fi folosit pentru o transmisie cu trimitere multiplă, cu vizualizatori care le acordează după dorinţă. Dacă toate cadrele depind de predecesoarele lor până la primul cadru, cineva care a pierdut primul cadru nu va putea decodifica cadrele succesive. În al doilea rând, dacă un cadru a fost recepţionat eronat, nu este posibilă decodificarea în continuare. În al treilea rând, fără cadre I, atunci când se face un avans sau o revenire rapidă, decodificatorul ar trebui să calculeze fiecare cadru peste care trece, aşa încât să ştie valoarea completă a cadrului pe care este oprit. Din aceste motive, cadrele I sunt inserate la ieşire o dată sau de două ori pe secundă.

Cadrele P, în contrast, codifică diferenţele între cadre. Ele se bazează pe ideea de **macroblocuri (macroblocks)**, care acoperă 16x16 pixeli în spaţiul luminanţei şi 8x8 pixeli în spaţiul crominanţei. Un macrobloc este codificat prin căutarea cadrului precedent sau ceva care diferă foarte puţin de el.

Un exemplu unde se folosesc cadrele P este redat în fig. 7-77. Aici vedem trei cadre consecutive care au acelaşi fundal, dar diferă prin poziţia unei persoane. Macroblocurile care conţin scena fundalului se vor potrivi exact, dar macroblocurile conținând persoana vor fi afișate la o poziție cu o deplasare de valoare necunoscută și va trebui să fie înregistrate.

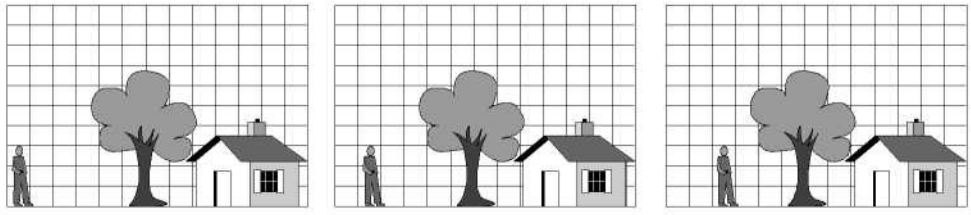


Fig. 7-77. Trei cadre consecutive.

Standardul MPEG-l nu specifică cum trebuie făcută căutarea, cât de departe să se caute sau cât de bună trebuie să fie o potrivire pentru a conta. Aceasta depinde de fiecare implementare. De exemplu, o implementare poate căuta un macrobloc la poziţia curentă din cadrul precedent şi toate celelalte deplasări ale poziţiilor ±Ax pe direcţia x şi ±Ay pe direcţia y. Pentru fiecare poziţie, poate fi calculat numărul de potriviri în matricea de luminanţă. Poziţia cu cel mai mare scor va fi declarată câştigătoare, cu condiţia să fi fost peste un prag predefinit. Altfel, macroblocul se spune că lipseşte. Sunt desigur posibili şi alţi algoritmi mai complicaţi.

Dacă este găsit un macrobloc, acesta este codificat luând diferenţa faţă de valoarea sa din cadrul anterior (pentru luminanţă şi pentru cele două crominanţe). Aceste matrice de diferenţe sunt apoi subiectul unei transformări cosinusoidale discrete, sunt cuantificate, codificate cu lungimea succesiunilor şi codificate Huffman, la fel ca şi cu JPEG. Valoarea pentru macrobloc în fluxul de ieşire este vectorul de mişcare (cât de departe s-a mişcat macroblocul din poziţia lui precedentă pe fiecare direcţie), urmată de lista de numere codificată Huffman. Dacă macroblocul nu este localizat în cadrul precedent, valoarea curentă este codificată cu JPEG, la fel ca în cadrul I.

Evident, algoritmul este puternic asimetric. O implementare este liberă să încerce fiecare poziţie plauzibilă din cadrul precedent, dacă vrea s-o facă, într-o încercare disperată de a localiza fiecare ultim macrobloc, oriunde s-ar fi mişcat acesta. Această tratare va minimiza fluxul MPEG-l codificat, cu preţul unei codificări foarte lente. Această abordare poate fi bună pentru o singură codificare a unei filmoteci, dar va fi groaznică pentru o video conferinţă în timp-real.

Similar, fiecare implementare este liberă să decidă ce înseamnă un macrobloc „găsit". Această libertate permite implementatorilor să concureze prin calitatea şi viteza algoritmilor proprii, dar întotdeauna produce MPEG-1. Indiferent de algoritmul de căutare folosit, ieşirea finală este ori codificarea JPEG a macroblocului curent, ori codificarea JPEG a diferenţei între macroblocul curent şi unul din cadrul precedent la o deplasare specificată faţă de cel curent.

Până acum, decodificarea MPEG-l este directă. Decodificarea cadrelor I este la fel ca decodificarea imaginilor JPEG. Decodificarea cadrelor P cere decodificatorului să memoreze cadrul precedent şi apoi să construiască noul cadru într-un nou tampon bazat pe macroblocuri codificate complet şi macroblocuri care conţin diferenţe faţă de cadrul precedent. Noul cadru este asamblat macrobloc cu macrobloc.

Cadrele B sunt similare cadrelor P, cu excepţia faptului că ele permit ca macroblocul de referinţă să fie sau într-un cadru precedent sau în cel următor. Această libertate suplimentară permite o compensare îmbunătăţită a mişcării şi este de asemenea utilă atunci când obiectele trec înaintea sau în spatele altor obiecte. Pentru a codifica cadrele B, codificatorul trebuie să ţină în memorie simultan trei cadre decodificate: cel vechi, actualul şi viitorul. Deşi cadrele B furnizează cea mai bună compresie, nu toate implementările le suportă.

Cadrele D sunt folosite doar pentru a face posibilă afişarea imaginilor de rezoluţie mică atunci când se face o revenire sau o înaintare rapidă. Realizarea decodificării MPEG-l normală în timp- real este destul de dificilă. Cerând ca decodificatorul s-o facă atunci când se mişcă prin video de zece ori mai repede decât normal, este prea mult. În schimb, cadrele D sunt folosite pentru a produce imagini de joasă rezoluţie. Fiecare intrare a cadrului D este valoarea medie a unui bloc, fără codificare ulterioară, făcând uşoară afişarea în timp-real. Această facilitate este importantă pentru a permite oamenilor să caute prin video, la viteză mare, o anumită scenă.

Terminând tratarea lui MPEG-l, să trecem la MPEG-2. Codificarea MPEG-2 este fundamental similară cu codificarea MPEG-l cu cadre I, P, şi B. Cadrele D nu sunt suportate. De asemenea, transformarea cosinusoidală discretă foloseşte un bloc de 10x10 în loc de 8x8, pentru a avea cu 50% mai mulţi coeficienţi şi implicit o calitate mai bună. Deoarece este destinat televiziunii şi DVD-ului, MPEG-2 suportă atât imagini progresive cât şi întreţesute, în timp ce MPEG-l suportă doar imagini progresive. Mai sunt şi alte detalii minore care diferă în cele două standarde.

În loc de a suporta un singur nivel de rezoluţie, MPEG-2 suportă patru: scăzut (352x240), principal (720x480), înalt-1440 (1440x1152) şi înalt (1920x1080). Rezoluţia scăzută este pentru VCR-uri şi compatibilitate cu MPEG-1. Principalul nivel este cel normal pentru difuzarea NTSC. Celelalte două sunt pentru HDTV.

Pentru o calitate foarte bună a ieşirii, MPEG-2 rulează în general la 4-8 Mbps.

7.4.8 Video la cerere

Mecanismul de video la cerere este câteodată comparat cu un magazin de închiriere a casetelor video. Utilizatorul (clientul) selectează una dintr-un număr mare de casete video pe care le are la dispoziţie şi o ia acasă pentru vizionare. Doar că pentru video la cerere, selecţia este făcută acasă folosind telecomanda televizorului şi caseta video începe imediat. Nu este necesar un drum până la magazin. Este inutil să spunem că implementarea video-ului la cerere este un pic mai complicată decât descrierea lui. În această secţiune vom face o prezentare a ideilor de bază şi a implementării lor.

Este video la cerere într-adevăr precum închirierea unei casete video, sau mai degrabă precum alegerea unui film pentru vizionare la sisteme de televiziune prin cablu cu 500 de canale? Răspunsul are importante implicaţii tehnice. În particular, utilizatorii de casete video închiriate sunt obişnuiţi cu ideea să oprească un film, să se ducă la bucătărie sau la baie şi apoi să continue din locul în care caseta video a fost oprită. Telespectatorii nu se aşteaptă să oprească un program.

Pentru a concura cu succes cu magazinele de închiriere, video-ul la cerere ar trebui să poată opri, porni şi relua casetele video la dorinţă. Asigurarea acestei posibilităţi obligă furnizorul de video să transmită o copie separată fiecărui utilizator.

Pe de altă parte, dacă video-ul la cerere este văzut mai mult ca o televiziune avansată, atunci este suficient ca furnizorul de casete video să pornească fiecare video popular la fiecare 10 minute şi să ruleze non-stop. Un utilizator care doreşte să vadă un astfel de film trebuie să aştepte cel mult 10 minute pentru ca el să înceapă. Deşi oprirea şi repornirea nu este posibilă aici, o persoană care se întoarce în cameră după o scurtă pauză, poate comuta pe un alt canal, care prezintă aceeaşi casetă video, dar cu o întârziere de 10 minute. Ceva se va repeta dar nu va fi pierdut nimic. Această schemă este numită **video aproape la cerere (near video on demand)**. El oferă posibilitatea unui cost mult mai scăzut, deoarece acelaşi semnal de la furnizorul de casete video poate ajunge la mai mulţi utilizatori odată. Diferenţa între video la cerere şi video aproape la cerere este similară cu diferenţa între a circula cu maşina proprie şi a lua autobuzul.

Urmărirea filmelor (aproape) la cerere este numai unul dintr-o gamă largă de noi servicii posibile de când este disponibilă reţeaua în bandă largă. Modelul general pe care mulţi îl folosesc este ilustrat în fig. 7-78. În centrul sistemului se află o reţea cu coloană vertebrală de arie largă (naţională sau internaţională) cu lărgime de bandă mare. La ea sunt conectate mii de reţele de distribuţie locală, cum ar fi cabluri TV sau sisteme distribuite ale companiei de telefoane. Sistemele de distribuţie locală ajung în casele oamenilor, unde se opresc în **cutii de conectare (set-top boxes)**, care sunt, de fapt, calculatoare personale specializate.

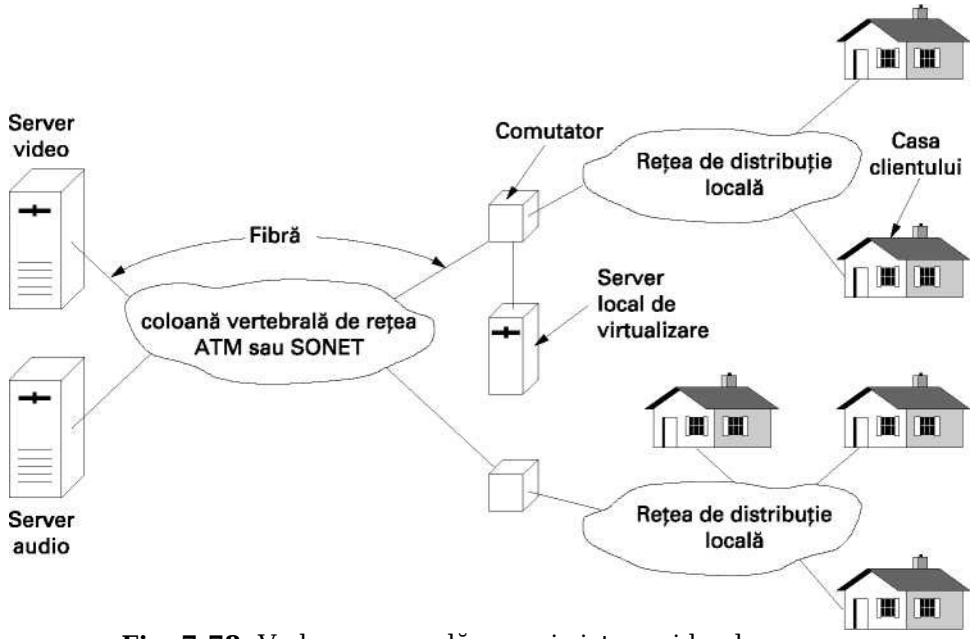


Fig. 7-78. Vedere generală a unui sistem video la cerere.

Mii de furnizori de informaţie sunt ataşaţi la coloana vertebrală prin fibre optice de lărgime de bandă mare. Unii dintre aceştia vor oferi video cu plata-pe-vizualizare sau CD audio cu plata-pe- ascultare. Altele vor oferi servicii specializate, precum cumpărături de la domiciliu (cu posibilitatea de a roti o cutie de supă şi a mări lista de ingrediente sau de a vedea un video-clip despre cum să conducă o maşină de cosit). Fără îndoială că în curând vor deveni disponibile sporturile, ştirile, reluările la „I Love Lucy”, accesul la WWW şi nenumăratele alte posibilităţi.

În sistem sunt incluse şi servere locale care permit ca video-urile să fie amplasate mai aproape de utilizatori (în avans), pentru a economisi lărgimea de bandă în timpul orelor de vârf. Cum se vor potrivi una cu alta aceste piese şi cui vor aparţine se va dezbate destul în industrie. În cele ce urmează, vom examina cum sunt create piesele principale din sistem: serverele video şi reţeaua de distribuţie.

Video-Servere

Pentru a avea video (aproape) la cerere, avem nevoie de **video-servere** (video servers) capabile să memoreze şi să transmită simultan un număr mare de filme. Numărul total de filme realizate este estimat la 65.000 (Minoli, 1995). Atunci când este comprimat în MPEG-2, un film normal ocupă în jur de 4 GB, aşa că 65.000 de filme ar necesita în jur de 260 teraocteţi. Adăugaţi la aceasta toate programele vechi de televiziune care au fost vreodată realizate, filmele de sport, jurnalele sonore, cataloagele vorbitoare pentru cumpărături etc. şi este clar că ne confruntăm cu o problemă de înmagazi- nare foarte dificilă.

Cea mai ieftină soluţie pentru memorarea unui volum mare de informaţie este pe bandă magnetică. Acesta a fost mereu cazul şi probabil va continua să fie. O bandă Ultrium poate memora 200 GB (50 filme) la un cost de circa 1 – 2 dolari/film.

Actualmente sunt comercializate servere mari, mecanice, de benzi care păstrează mii de benzi şi au un braţ robotic pentru extragerea fiecărei benzi şi inserarea ei într-o unitate de bandă. Problema cu aceste sisteme este timpul de acces (în special pentru al 50-lea film de pe bandă), viteza de transfer şi numărul limitat de unităţi de bandă (pentru a servi *n* filme deodată, unitatea necesită *n* unităţi).

Din fericire, experienţa cu magazinele de închiriat video, bibliotecile publice şi alte astfel de organizaţii arată că nu toate produsele sunt la fel de populare. Experimental, atunci când sunt disponibile *N* filme, fracţia tuturor cererilor pentru filmul care ocupă locul *k* în topul popularităţii este de aproximativ *C/k.* Aici *C* este calculat pentru a normaliza suma la 1, cu formula:

$$C = 1/(1 + 1/2 + 1/3 + 1/4 + 1/5 + \ldots + 1/N)$$

Astfel, filmul de pe primul loc este de şapte ori mai popular decât filmul de pe locul şapte. Acest rezultat este cunoscut drept **legea lui Zipf** (Zipf, 1949).

Faptul că unele filme sunt mai populare decât altele sugerează o soluţie posibilă în forma ierarhiei de memorare, aşa cum se arată în fig. 7-79. Aici, performanţa creşte cu cât se urcă în ierarhie.

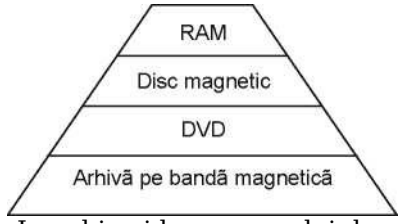


Fig. 7-79. Ierarhia video-serverului de memorare.

O alternativă la înregistrarea pe bandă este memoria optică. DVD-urile actuale memorează 4.7 GB, suficienţi pentru un film, dar următoarea generaţie va păstra două filme. Deşi timpii de căutare sunt mici în comparaţie cu discurile magnetice (50 ms faţă de 5 ms), costul lor scăzut şi fiabilitatea ridicată fac din tonomatele optice care conţin mii de DVD-uri o alternativă bună faţă de bandă în cazul filmelor cel mai mult folosite.

Urmează discurile magnetice. Acestea au timp de acces mic (5 ms), viteza de transfer mare (320 MB/sec pentru SCSI 320), şi capacităţi substanţiale (> 100 GB), care le fac potrivite pentru memorarea filmelor care sunt transmise efectiv (spre deosebire de simpla memorare pentru cazul că cineva ar dori vreodată să le vadă). Marele lor inconvenient este costul ridicat pentru memorarea filmelor care sunt rar accesate.

În vârful piramidei din fig. 7-79 este RAM. RAM-ul este cel mai rapid mediu de stocare, dar şi cel mai scump. Atunci când preţurile la RAM ajung la 50 dolari/gigaoctet, un film de 4 GB va ocupa RAM în valoare de 200 dolari, aşa că, existenţa a 100 de filme în RAM va costa 20.000 dolari pentru 200 GB de memorie. În plus, ideea unui video-server care transmite 100 de filme, păstrându-le pe toate în RAM, începe să devină realizabilă. Şi dacă video-serverul are 100 de clienţi, dar ei se uită simultan doar la 20 de filme diferite, ideea începe să pară realizabilă,

având un concept bun.

Deoarece un video-server nu este decât un imens dispozitiv de I/O în timp-real, el necesită o altă arhitectură hardware şi software decât un PC sau o staţie de lucru UNIX. Arhitectura hardware a unui video-server tipic este prezentată în fig. 7-80. Serverul are una sau mai multe unităţi centrale de înaltă performanţă, fiecare cu memorie locală, o memorie principală partajată, o memorie tampon RAM de mare capacitate pentru filmele populare, o varietate de echipamente de stocare pentru păstrarea filmelor şi hardware de conectare în reţea, în mod normal o interfaţă optică cu o reţea SONET sau ATM la viteza unui OC-12 sau mai ridicată. Aceste subsisteme sunt conectate printr-o magistrală de foarte mare viteză (cel puţin de 1 GB/sec).

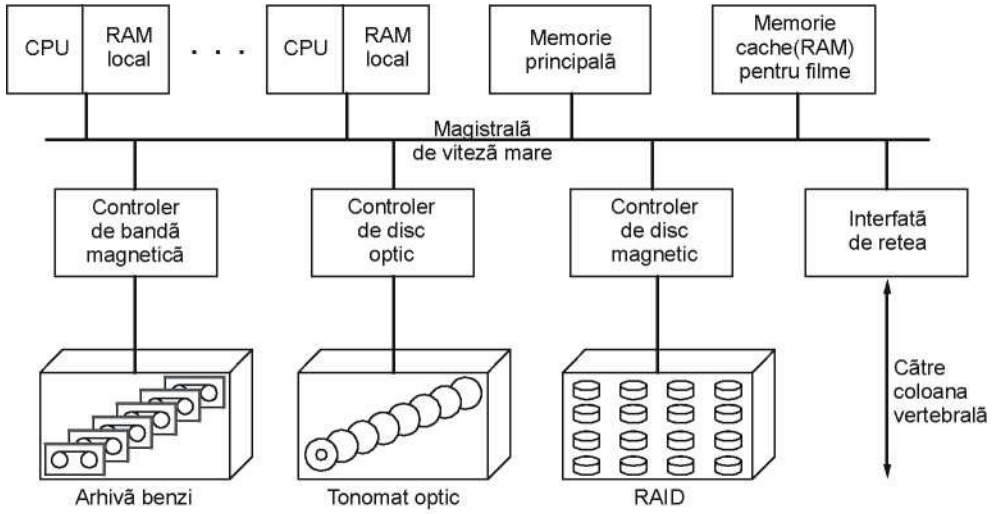


Fig. 7-80. Arhitectura hardware a unui video-server tipic.

Acum să inspectăm programele pentru video-server. Unităţile centrale sunt folosite pentru acceptarea cererilor utilizatorilor, localizarea filmelor, mutarea datelor între echipamente, taxarea clienţilor, şi multe alte funcţii. Unele dintre acestea nu sunt critice în timp, dar multe altele sunt, aşa că unele, dacă nu toate unităţile centrale, va trebui să ruleze un sistem de operare de timp-real, cum ar fi un micronucleu de timp-real. În mod normal aceste sisteme împart munca în operaţii mai simple, fiecare cu un termen final cunoscut. Atunci planificatorul poate rula un algoritm de tip „alege termenul cel mai apropiat” sau un algoritm monoton al vitezelor (Liu şi Layland, 1973).

Software-ul unităţii centrale defineşte şi natura interfeţei pe care serverul o prezintă clienţilor (servere de virtualizare şi cutii de conectare). Există două tipuri populare. Primul este un sistem de fişiere tradiţional, în care clienţii pot deschide, citi, scrie şi închide fişiere. Spre deosebire de complicaţiile introduse de ierarhia de memorii şi consideraţiile de timp-real, un astfel de server poate avea un sistem de fişiere modelat după cel din UNIX.

Al doilea tip de interfaţă este bazat pe modelul video-recorderului. Comenzile

adresate serverului solicită deschiderea, rularea, oprirea, derularea rapidă înainte şi înapoi a fişierelor. Diferenţa faţă de modelul UNIX este că atunci când este dată o comandă PLAY, serverul continuă să transmită date la o viteză constantă, fără a necesita comenzi noi.

Inima software-ului video-serverului este sistemul de gestiune a discului. Acesta are două misiuni principale: plasarea filmelor pe discul magnetic atunci când trebuie extrase de pe bandă sau din memoria optică şi tratarea cererilor către disc pentru multiplele fluxuri de ieşire. Plasarea filmelor este importantă, deoarece poate afecta foarte mult performanţele.

Două modalităţi de organizare a memorării pe disc sunt ferma de discuri şi şirul de discuri. În cazul **fermei de discuri** (disk farm), fiecare unitate păstrează doar câteva filme întregi. Din motive de performanţă şi fiabilitate, fiecare film trebuie să fie prezent pe cel puţin două unităţi, poate chiar mai multe. Cealaltă organizare de memorare este cea de **tablou de discuri** (disk array) sau **RAID** (**Redundant Array of Inexpensive Disks –** tablou redundant de discuri ieftine), în care fiecare film este împărţit pe mai multe unităţi, de exemplu blocul 0 pe unitatea 0, blocul 1 pe unitatea 1, şi aşa mai departe, cu blocul n – 1 pe unitatea n – 1. După aceasta, ciclul se repetă, cu blocul n pe unitatea 0 şi aşa mai departe. Această organizare este numită **repartizare** (striping).

Un tablou de discuri repartizat are mai multe avantaje decât o fermă de discuri. Mai întâi, toate cele n unităţi pot rula în paralel, mărind performanţa cu factorul n. În al doilea rând, poate fi făcut redundant prin adăugarea unei unităţi în plus la fiecare grup de n, unde unitatea redundantă conţine SAU EXCLUSIV bloc-cu-bloc ale celorlalte unităţi, pentru a permite recuperarea completă a datelor în cazul în care o unitate se defectează. În sfârşit, problema echilibrării încărcării este rezolvată (nu este necesară plasarea manuală pentru evitarea plasării tuturor filmelor populare pe aceeaşi unitate). Pe de altă parte, organizarea de tip tablou de discuri este mai complicată decât ferma de discuri şi mai sensibilă la defecte multiple. De asemenea este nepotrivită pentru operaţii ale video- recorder-ului, cum ar fi derularea rapidă a unui film înainte sau înapoi.

Cealaltă misiune a software-ului de disc este de a servi toate fluxurile de ieşire de timp-real şi de a respecta constrângerile de timp ale acestora. Doar acum câţiva ani, aceasta necesita algoritmi complecşi de planificare a sarcinilor, dar odată cu scăderea preţurilor la memorie, încep să devină posibile abordări mult mai simple. Pentru fiecare flux servit, este păstrată în RAM o zonă tampon (buffer) de, să zicem, 10 secunde de flux video (care înseamnă un spaţiu ocupat de 5 MB). El este completat de un proces al discului şi golit de un proces al reţelei. Cu 500 MB de RAM, pot fi servite 100 de fluxuri direct din RAM. Desigur, subsistemul discului trebuie să aibă o rată susţinută de 50MB/sec pentru a păstra zonele tampon pline, dar un RAID construit din discuri SCSI de ultimă generaţie poate să îndeplinească uşor această cerinţă.

Reţeaua de distribuţie

Reţeaua de distribuţie este un set de comutatoare şi linii între sursă şi destinaţie. Aşa cum vedem în fig. 7-78, ea constă dintr-o coloană vertebrală

conectată la o reţea de distribuţie locală. În mod obişnuit, coloana vertebrală este comutată, dar reţeaua locală nu.

Principala cerinţă impusă coloanei vertebrale este lărgimea de bandă mare. O altă cerinţă era ca fluctuaţia să fie scăzută, însă acum, chiar şi cu cel mai mic PC este posibilă stocarea într-un tampon a 10 secunde de video de înaltă calitate MPEG-2 şi prin urmare, fluctuaţia scăzută nu mai este o necesitate.

Distribuţia locală este haotică, diferite companii încercând diferite reţele în diferite regiuni. Companiile telefonice, companiile de TV prin cablu şi noii intraţi sunt convinşi cu toţii că cel care ajunge primul va fi câştigătorul cel mare. În consecinţă asistăm la o proliferare a tehnologiilor instalate. În Japonia, unele companii de canalizare au intrat în afacerea Internet, susţinând că ele au cele mai mari ţevi în casele tuturor (introduc fibră optică prin ele, dar trebuie să fie foarte atente pe unde o scot). Cele patru scheme principale de distribuţie locală pentru video la cerere sunt identificate prin acronimele ADSL, FTTC, FTTH şi HFC. Le vom explica pe fiecare pe rând.

ADSL a fost primul reprezentant al industriei telefonice în loteria distribuţiei locale. Am studiat ADSL în cap. 2 şi nu vom repeta acel material aici. Ideea este că fiecare casă din Statele Unite, Europa şi Japonia are deja o pereche torsadată de cupru (pentru servicii telefonice analogice). Dacă aceste fire pot fi folosite pentru video la cerere, companiile telefonice ar putea să elimine concurenţa.

Problema, desigur, este că aceste fire nu pot suporta nici chiar MPEG-l pe lungimea lor tipică de 10 km, ca să nu mai vorbim de MPEG-2. Filmele color, de înaltă rezoluţie, necesită 4-8 Mbps, depinzând de calitatea dorită. ADSL nu este suficient de rapid decât pentru bucle locale scurte.

Al doilea proiect al companiei telefonice este **FTTC** (**Fiber To The Curb –** fibră către vecinătate). În FTTC, compania telefonică instalează fibră optică de la oficiul final la fiecare cartier rezidenţial, terminată într-un echipament numit **ONU** (**Optical Network Unit –** unitate optică de reţea). Cele 16 bucle locale de cupru se pot termina în ONU. Aceste bucle sunt acum atât de scurte, încât este posibilă rularea duplex integrală T1 sau T2 peste ele, permiţând filmele MPEG-l şi respectiv MPEG-2. În plus, deoarece FTTC este simetric, acum este posibilă video conferinţa pentru cei care lucrează acasă şi pentru întreprinderile mici.

A treia soluţie a companiei telefonice este de a introduce fibra optică în casele tuturor. Se numeşte **FTTH** (**Fiber To The Home –** fibră la casă). În această schemă, oricine poate avea OC-l, OC- 3, sau chiar un purtător mai performant, dacă este cerut. FTTH este foarte scump, dar va deschide o gamă largă de posibilităţi atunci când va fi introdus. În fig. 7-63 am văzut cum oricine ar putea să aibă propriul său post de radio. Ce-aţi zice de ideea ca fiecare membru al familiei să aibă propriul post de televiziune? ADSL, FTTC şi FTTH sunt toate reţele locale de distribuţie punct-la-punct, ceea ce nu este surprinzător dată fiind organizarea actuală a sistemului telefonic.

O abordare complet diferită este **HFC** (**Hybrid Fiber Coax –** fibră coaxială hibridă), care este soluţia preferată actualmente, fiind instalată în prezent de către firmele de televiziune prin cablu. Aceasta este prezentată în Fig. 2-47(a). Povestea este următoarea. Cablurile coaxiale actuale de la 300 la 450 MHz vor fi înlocuite

prin cabluri coaxiale la 750 MHz, îmbunătăţind capacitatea de la 50 la 75 canale de 6 MHz la 125 canale de 6-MHz. Şaptezeci şi cinci din cele 125 de canale vor fi folosite pentru transmiterea televiziunii analogice.

Cele 50 de canale noi vor fi modulate folosind QAM-256, care furnizează în jur de 40 Mbps pe canal, dând un total de 2 Gbps de lărgime de bandă nouă. Capetele vor fi mutate în cartier, aşa încât fiecare cablu este doar pentru 500 de case. Simpla împărţire arată că fiecărei case îi poate fi alocat un canal dedicat de 4 Mbps, care poate fi folosit pentru un film MPEG-2.

Deşi sună minunat, cere furnizorilor de cabluri să le înlocuiască pe cele existente cu cabluri coaxiale de 750 MHz, să instaleze noile capete de distribuţie şi să elimine toate amplificatoarele unidirecţionale – pe scurt, să înlocuiască întregul sistem de TV prin cablu. În consecinţă, volumul de infrastructură nouă este comparabil cu ceea ce este necesar companiilor telefonice pentru FTTC. În ambele cazuri, furnizorul reţelei locale trebuie să instaleze fibra optică în cartierele rezidenţiale. Din nou, în ambele cazuri, fibra se termină la un convertor optic-electric. În FTTC, segmentul final este o buclă locală punct-la-punct care foloseşte perechi torsadate. În HFC, segmentul final este un cablu coaxial partajat. Tehnic vorbind, aceste două sisteme nu sunt chiar atât de diferite pe cât vor să le prezinte creatorii lor.

Cu toate acestea, există o diferenţă reală care merită să fie amintită. HFC foloseşte un mediu partajat fără comutare sau dirijare. Orice informaţie transmisă prin cablu poate fi preluată de orice abonat fără multă zarvă. FTTC, care este complet comutat, nu are această proprietate. Ca rezultat, operatorii HFC vor ca video-serverele să trimită fluxuri criptate, aşa încât clienţii care nu au plătit pentru un film, să nu-l poată vedea. Operatorii FTTC nu doresc criptarea deoarece măreşte complexitatea, scade performanţa şi nu furnizează securitate suplimentară în sistemul lor. Din punctul de vedere al unei companii care rulează un video-server, este o bună idee să se cripteze sau nu? Un server folosit de o companie telefonică sau unul din subsidiarii sau partenerii săi poate să decidă să nu cripteze video-urile, pretinzând eficienţa ca motiv, dar de fapt pentru a cauza pierderi economice competitorilor HFC.

Pentru toate reţelele locale de distribuţie, este posibil că fiecare cartier va fi echipat cu unul sau mai multe servere de virtualizare. Acestea sunt, de fapt, doar versiuni mai mici ale video-serverelor despre care am discutat înainte. Marele avantaj al acestor servere locale este că reduc încărcarea coloanei vertebrale.

Ele pot fi preîncărcate cu filme fie dinamic, fie prin rezervare. Dacă oamenii spun furnizorului în avans ce filme doresc, ele pot fi transferate pe serverul local în afara orelor de vârf. Această observaţie orientează operatorii de reţea spre atragerea personalului de la companiile aeriene pentru stabilirea tarifelor. Se pot imagina tarife în care filmele cerute cu 24 până la 72 de ore în avans pentru a fi vizionate marţea sau joia înainte de 6 seara sau după 11 seara, primesc o reducere de 27 la sută. Filmele comandate în prima duminică a lunii înainte de 8 dimineaţa pentru a fi vizionate miercuri după-amiaza, într-o zi a cărei dată este un număr prim, beneficiază de o reducere de 43 la sută şi aşa mai departe.

7.4.9 MBone – Coloana vertebrală pentru trimitere multiplă

În timp ce toate aceste industrii fac planuri mari – şi îndelung mediatizate – pentru viitorul video la cerere (inter)naţional, digital, comunitatea Internet şi-a implementat propriul sistem multimedia digital, **MBone (Multicast Backbone –** coloana vertebrală cu trimitere multiplă**)**. În această secţiune vom face o scurtă sinteză a ceea ce este şi cum funcţionează.

MBone poate fi gândit ca radio şi televiziune Internet. Spre deosebire de video la cerere, unde accentul cade pe selectarea şi vizualizarea filmelor precomprimate memorate pe un server, MBone este folosit pentru difuzare audio şi video în formă digitală în lumea întreagă prin Internet. Este operaţional de la începutul lui 1992. Multe conferinţe ştiinţifice, în special întâlniri IETF, au fost difuzate, la fel ca şi evenimentele ştiinţifice notabile, cum ar fi lansarea navetelor spaţiale. Prin MBone a fost difuzat un concert Rolling Stones, precum şi porţiuni din Festivalul de film de la Cannes. Este discutabil dacă acesta poate fi calificat drept un eveniment ştiinţific.

Din punct de vedere tehnic, MBone este o reţea virtuală situată deasupra Internet-ului. Ea constă din insule cu posibilităţi de trimitere multiplă, conectate prin tuneluri, aşa cum se arată în fig. 781. În această figură, MBone constă din şase insule, de la *A* la *F,* conectate prin şapte tuneluri. Fiecare insulă (de obicei un LAN sau un grup de LAN-uri interconectate) suportă trimitere multiplă hardware către calculatoarele gazdă. Tunelurile propagă pachetele MBone între insule. Cândva, în viitor, când toate ruterele vor fi capabile să gestioneze direct traficul cu trimitere multiplă, această superstructură nu va mai fi necesară, dar pentru moment este funcţională.

Fiecare insulă conţine unul sau mai multe rutere specializate numite **m-rutere** (**mrouters –** rutere cu trimitere multiplă). Câteva dintre acestea sunt rutere normale, dar majoritatea sunt numai staţii UNIX care rulează software-ul special de nivel utilizator (dar ca supervizor). M-ruterele sunt conectate logic prin tuneluri. Pachetele MBone sunt încapsulate în pachete IP şi trimise ca pachete obişnuite cu trimitere unică la adresa IP a m-ruterului destinaţie.

Tunelurile sunt configurate manual. În mod uzual, un tunel este o cale pentru care există o conexiune fizică, dar aceasta nu este o cerinţă. Dacă, accidental, calea fizică asociată unui tunel se defectează, m-ruterele care folosesc tunelul nu vor observa, deoarece Internet-ul va redirija automat întregul trafic IP dintre ele prin alte linii.

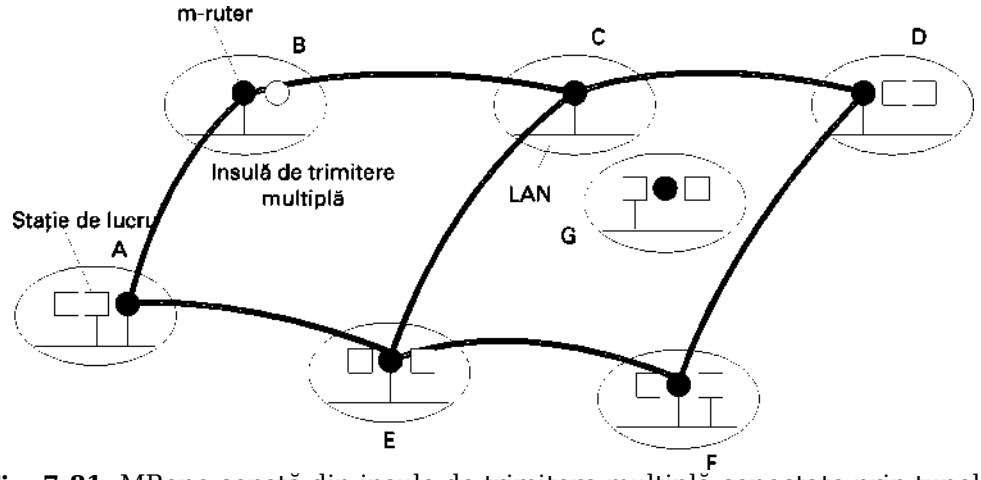


Fig. 7-81. MBone constă din insule de trimitere multiplă conectate prin tunele.

Atunci când apare o nouă insulă şi doreşte să se ataşeze la MBone, precum *G* din fig. 7-81, administratorul său trimite un mesaj anunţând existenţa acesteia către lista de poştă a MBone-ului. Administratorii siturilor apropiate îl contactează pentru a stabili tunelurile. Câteodată tunelurile existente sunt reconfigurate, astfel încât să profite de noua insulă pentru a optimiza topologia. La urma urmelor, tunelurile nu au existenţă fizică. Ele sunt definite prin tabele în m-rutere şi pot fi adăugate, şterse, sau mutate prin simpla schimbare a acestor tabele. Tipic, fiecare ţară din MBone are o coloană vertebrală cu insule regionale ataşate acesteia. În mod normal, MBone este configurat cu unul sau două tuneluri care traversează oceanele Atlantic şi Pacific, aducând MBone-ul la scară globală.

Astfel, în orice moment, MBone constă dintr-o topologie specifică alcătuită din insule şi tuneluri, independent de numărul adreselor de trimitere multiplă utilizate curent şi de cine le ascultă sau le urmăreşte. Această situaţie este foarte asemănătoare cu cea a unei subreţele normale (fizice), aşa încât i se aplică algoritmii normali de dirijare. În consecinţă, MBone a folosit iniţial un algoritm de dirijare, **DVMRP (Distance Vector Multicast Routing Protocol –** dirijare multi-destinaţie după vectorul distanţelor**)** bazat pe algoritmul vectorului distanţelor al lui Bellman-Ford. De exemplu, în fig. 781, insula *C* poate dirija către *A* prin *B* sau prin *E* (sau prin *D).* Îşi alege varianta luând valorile pe care i le dau nodurile despre distanţele de la ele până la *A* şi apoi adăugând distanţa proprie până la ele. În acest mod, fiecare insulă determină ruta optimă către fiecare altă insulă. Totuşi, rutele nu sunt de fapt folosite în acest mod, ci aşa cum vom vedea în curând.

Să considerăm acum modul în care se realizează trimiterea multiplă. Pentru a transmite la mai multe destinaţii un program audio sau video, o sursă trebuie să achiziţioneze mai întâi o adresă de destinaţie multiplă de clasă D, care acţionează ca o frecvenţă de staţie sau un număr de canal. Adresele de clasă D sunt rezervate prin folosirea unui program care caută într-o bază de date adrese de destinaţie multiplă libere. Multe trimiteri multiple pot să se desfăşoare în acelaşi timp, iar un calculator gazdă poate „să se acordeze” pe cea de care este interesat prin

ascultarea adrese de destinaţie multiplă potrivite.

Periodic, fiecare m-ruter trimite un pachet de difuzare IGMP limitat la insula lui, întrebând cine este interesat de ce canal. Calculatoarele gazdă care doresc să (continue să) primească unul sau mai multe canale trimit alt pachet IGMP înapoi drept răspuns. Aceste răspunsuri sunt dispersate în timp pentru a evita supraîncărcarea LAN-ului local. Fiecare m-ruter păstrează o tabelă cu canalele care trebuie puse în LAN-ul propriu, pentru a evita pierderea de lărgime de bandă prin canale de transmitere multiplă pe care nu le vrea nimeni.

Trimiterile multiple se propagă prin MBone după cum urmează. Când o sursă audio sau video generează un nou pachet, îl difuzează către insula sa locală folosind facilităţile hardware de trimitere multiplă. Acest pachet este preluat de un m-ruter local, care îl copiază pe toate tunelurile cu care este conectat.

Fiecare m-ruter care primeşte un astfel de pachet printr-un tunel verifică dacă pachetul a venit pe cea mai bună rută, adică cea pe care tabela proprie îi spune să o folosească pentru a ajunge la sursă (ca şi cum ar fi o destinaţie). Dacă pachetul a venit pe ruta cea mai bună, m-ruterul îl copiază la toate celelalte tuneluri. Dacă pachetul a ajuns pe o cale care nu este cea optimală, el este eliminat. Astfel, de exemplu, în fig. 7-81, dacă tabela lui *C* spune să se folosească *B* pentru a ajunge la *A,* atunci când un pachet cu trimitere multiplă de la *A* ajunge la *C* prin *B,* el este copiat la *D* şi *E.* Cu toate acestea, atunci când un pachet cu trimitere multiplă de la *A* ajunge la *C* prin *E* (nu este calea cea bună), el este eliminat. Acest algoritm este algoritmul retransmiterii pe calea inversă prezentat în Cap. 5. Deşi nu este perfect, este destul de bun şi foarte uşor de implementat.

În plus faţă de folosirea algoritmului de căutare pe calea inversă, pentru prevenirea inundării În- ternet-ului şi pentru a limita domeniul trimiterii multiple, este folosit câmpul IP *Time_to_live* (timp de viaţă). Fiecare pachet pleacă cu o anumită valoare (determinată de sursă). Fiecărui tunel i se asociază o pondere. Un pachet este trecut printr-un tunel dacă are o pondere suficientă. Altfel este eliminat. De exemplu, tunelurile transoceanice sunt configurate în mod obişnuit cu o pondere de 128, aşa încât pachetele pot fi limitate la continentul de origine dându-li-se un timp de viaţă iniţial mai mic sau egal cu 127. După trecerea printr-un tunel, câmpul *Time to live* este decrementat cu ponderea tunelului.

De când funcţionează algoritmul de dirijare MBone, s-au făcut multe cercetări pentru a-l îmbunătăţi. O propunere păstrează ideea dirijării după vectorul distanţelor, dar face algoritmul ierarhic, prin gruparea siturilor MBone în regiuni şi dirijarea în prima etapă către acestea (Thyagarajan şi Deering, 1995).

O altă propunere este de a folosi o formă modificată a dirijării în funcţie de starea legăturilor, în loc de dirijare după vectorul distanţelor. În particular, un grup de lucru IETF se ocupă de modificarea OSPF pentru a-l face potrivit pentru trimitere multiplă în cadrul unui singur sistem autonom. Trimiterea multiplă OSPF rezultată este numită **MOSPF** (Moy, 1994). Modificările se referă la crearea de către MOSPF a unei hărţi complete care, în plus faţă de informaţia vizuală de dirijare, să ţină evidenţa insulelor de trimitere multiplă şi a tunelurilor. Înarmată cu această topologie completă, este uşor să calculăm cea mai bună cale de la fiecare insulă către fiecare altă insulă folosind tunelurile. De exemplu, poate fi folosit

algoritmul lui Dijkstra.

A doua arie de cercetare este dirijarea inter-AS. Aici un alt grup de lucru IETF dezvoltă un algoritm numit **PIM (Protocol Independent Multicast –** transmitere multiplă independentă de protocol**)**. PIM are două versiuni, după cum insulele sunt dense (aproape oricine vrea să se uite) sau rare (aproape nimeni nu vrea să se uite). Ambele versiuni folosesc tabele de dirijare standard cu trimitere unică în loc de a crea o topologie suprapusă, aşa cum fac DVMRP sau MOSPF.

În PIM-DM (modul dens), ideea este de a tăia căile inutile. Tăierea funcţionează în modul următor. Atunci când un pachet cu trimitere multiplă ajunge printr-un tunel „greşit”, un pachet de tăiere este trimis înapoi prin tunel, spunând emiţătorului să nu-i mai transmită pachete de la sursa

respectivă. Când un pachet ajunge prin tunelul „bun”, este copiat pe toate celelalte tuneluri care nu s-au auto-tăiat anterior. Dacă toate celelalte tuneluri s-au auto-tăiat şi canalul din insula locală nu este interesat, m-ruterul trimite un mesaj de tăiere înapoi prin canalul „bun”. În acest fel, trimiterea multiplă se adaptează automat şi merge doar unde este dorită.

PIM-SM (modul rar), descris în RFC 2362, lucrează diferit. Aici ideea este de a preveni saturarea Internetului, doar pentru că trei persoane din Berkeley vor să ţină o conferinţă peste o adresă de clasă D. PIM-SM funcţionează prin fixarea unor puncte de întâlnire. Fiecare dintre sursele dintr-un grup cu trimitere multiplă PIM-SM îşi trimite pachetele la punctele de întâlnire. Orice sit interesat în ataşare, cere unui punct de întâlnire să-i seteze un tunel. În acest mod, tot traficul PIM-SM este transportat prin transmitere simplă, în loc de transmitere multiplă. PIM-SM devine tot mai popular şi MBONE migrează către folosirea lui. Pe măsură ce PIM-SM devine mai mult folosit, MOSPF dispare treptat. Pe de altă parte, însuşi MBONE pare într-un fel să stagneze şi probabil niciodată nu va deveni foarte popular.

Totuşi, multimedia prin reţea este încă un domeniu foarte interesant, care evoluează rapid, chiar dacă MBONE nu devine un succes uriaş. Zilnic sunt anunţate noi tehnologii şi aplicaţii. Mai mult, transmiterea multiplă şi calitatea serviciului funcţionează împreună, după cum se prezintă în (Striegel şi Manimaran, 2002). Alt subiect fierbinte este transmisia multiplă fără fir (wireless) (Gos- sain et. al., 2002). Întregul domeniu al transmisiunilor multiple şi orice este legat de el va rămâne, probabil, important pentru următorii ani.

7.5 REZUMAT

Atribuirea numelor în Internet foloseşte o schemă ierarhică, numită sistemul numelor de domenii (DNS). La nivelul superior, există bine cunoscutele domenii generice, incluzând *com* şi *edu,* precum şi cele aproximativ 200 domenii pentru ţări. DNS este implementat ca un sistem de baze de date distribuite, cu servere în întreaga lume. DNS păstrează înregistrări cu adrese IP, centre de mesagerie şi alte informaţii. Prin interogarea unui server DNS, un proces poate stabili corespondenţa dintre un nume de domeniu Internet şi o adresă IP folosită pentru a comunica cu acel domeniu.

E-mail este una din cele două aplicaţii foarte populare din Internet. Oricine, de la copii la bunici, o poate folosi acum. Cele mai multe sisteme de poştă electronică din lume folosesc sistemul definit în RFC 2821 şi 2822. Mesajele trimise în acest sistem folosesc antete de sistem ASCII pentru definirea proprietăţilor mesajului. Materiale cu diverse tipuri conţinut pot fi transmise folosind MIME. Mesajele sunt transmise folosind SMTP, care lucrează făcând o conexiune TCP de la sistemul sursă la cel de destinaţie şi livrând în mod direct e-mail-ul peste conexiunea TCP.

O altă aplicaţie foarte populară pentru Internet este World Wide Web. Web-ul este un sistem pentru legarea documentelor "hipertext". Iniţial, fiecare document

era o pagină scrisă în HTML, cu posibile hiper-legături la alte documente. Azi, XML câştigă teren în faţa HTML. De asemenea, un mare volum de informaţie este generat dinamic, folosind script-uri executate de server (eng.:server- side scripts) (PHP, JSP şi ASP), precum şi script-uri executate de client (eng.: client-side scripts) (de remarcat aici Javascript). Un program de navigare poate afişa documentul stabilind o conexiune TCP cu serverul său, cerând documentul şi apoi închizând conexiunea. Aceste mesaje de cerere conţin o mulţime de antete pentru asigurarea informaţiei suplimentare. Folosirea memoriei ascunse, replicarea şi reţelele de livrare a conţinutului sunt folosite pe scară largă pentru a îmbunătăţi performanţele Web-ului.

Web-ul fără fir este de-abia la început. Primele sisteme sunt WAP şi i-mode, fiecare cu ecrane mici şi lungime de bandă limitate, dar cele din generaţia următoare vor fi mai puternice.

Multimedia este şi ea o stea pe firmamentul reţelelor. Se permite ca semnalele video şi audio să fie digitizate şi transportate electronic pentru afişare. Audio necesită mai puţină lărgime de bandă, astfel că este mai avansat. Fluxurile audio, radio prin Internet şi vocea prin IP (voice over IP) sunt acum o realitate, iar noile aplicaţii apar permanent. Video la cerere este un domeniu de viitor, de mare interes. În sfârşit, Mbone este un serviciu experimental, bazat pe televiziunea şi radioul digital trimise peste Internet.

7.6 PROBLEME

1. Multe calculatoare ale unor firme au trei identificatori universali, unici. Care sunt ei?

2. După informaţiile date în fig. 7-3, *little-sister.cs.vu.nl* se află într-o reţea de clasă A, B, sau C?

3. În fig. 7-3, nu este nici un punct după *rowboat?* De ce nu?

4. Ghiciţi ce ar putea să însemne smiley-ul :-X (uneori scris ca :-#).

5. DNS foloseşte UDP în loc de TCP. Pachetele DNS pierdute nu pot fi recuperate automat. Cauzează acest lucru probleme, şi dacă da, cum sunt ele rezolvate?

6. În plus faţă de problema pierderilor, pachetele UDP au o dimensiune maximă, uneori ajungând chiar la minimum 576 octeţi. Ce se întâmplă când numele DNS căutat depăşeşte această dimensiune? Poate fi trimis în două pachete?

7. Se poate ca o maşină cu un singur nume DNS să aibă mai multe adrese IP? Cum ar putea să se întâmple acest lucru?

8. Este posibil ca un calculator să aibă două nume DNS care aparţin de două domenii de nivel înalt? Dacă da, daţi un exemplu plauzibil. Daca nu, explicaţi de

ce.

9. Numărul de companii cu site Web a crescut exploziv în ultimii ani. Ca rezultat, mii de companii au fost înregistrate în domeniul *com,* ducând la o încărcare mare a serverului pentru acest domeniu. Sugeraţi o cale de a diminua această problemă fără a schimba schema de nume (adică fără a introduce noi domenii de nivel înalt). Este permis ca soluţia să impună schimbarea codului de la client.

10. Unele sisteme de e-mail suportă în antet câmpul *Content Return:.* El specifică dacă corpul mesajului trebuie să fie returnat în cazul imposibilităţii livrării. Acest câmp aparţine plicului sau conţinutului?

11. Sistemele de poştă electronică au nevoie de registre pentru a putea căuta adresele de e-mail. Pentru a construi asemenea registre, numele ar trebui să fie separate în componentele standard (de exemplu nume, prenume) pentru a putea fi făcute căutări. Discutaţi unele dintre problemele ce trebuie rezolvate pentru acceptarea universală a unui astfel de standard.

12. Adresa de e-mail a unei persoane este numele său de utilizator @ numele DNS cu o înregistrare MX. Numele de utilizator pot fi prenume, nume, iniţiale, tot felul de alte nume. Să presupunem că o firmă mare a decis că se pierdea prea mult e-mail din cauză că lumea nu cunoştea numele de utilizator al destinatarului. Există vreo modalitate pentru ca ei să rezolve această problemă fără schimbarea DNS-ului? Dacă da, daţi o propunere şi explicaţi cum funcţionează. Dacă nu, explicaţi de ce este imposibil.

13. Un fişier binar are lungimea de 3072 de biţi. Cât de lung va fi dacă îl codificăm folosind base64, cu perechea CR+LF inserată după fiecare 80 de octeţi transmişi şi la sfârşit?

14. Consideraţi schema de codificare MIME afişabilă-marcată. Menţionaţi o problemă nediscutată în text şi propuneţi o soluţie.

15. Daţi cinci tipuri MIME care nu sunt listate în text. Puteţi să verificaţi browser-ul dvs. sau să cău- taţi pe Internet.

16. Să presupunem că vreţi să trimiteţi un fişier MP3 la un prieten, dar ISP-ul prietenului limitează dimensiunea unui mesaj la 1 MB iar fişierul MP3 are 4 MB. Există vreo modalitate de a rezolva problema aceasta folosind RFC 822 şi MIME?

17. Să presupunem că cineva instalează un demon de vacanţă (vacation daemon) şi apoi trimite un mesaj chiar înainte de a ieşi din sistem. Din păcate, destinatarul este în vacanţă de o săptămână şi are de asemenea instalat un demon de vacanţă. Ce se întâmplă în continuare? Replicile vor fi trimise dintr-o parte în alta până când se va întoarce cineva?

18. În orice standard, ca de exemplu RFC 822, este necesară o gramatică precisă a ceea ce este permis astfel încât implementări diferite să poată conlucreze. Chiar

şi unităţile simple trebuie să fie definite cu atenţie. Antetele SMTP permit existenţa spaţiului între simboluri. Daţi *două* definiri plauzibile ale spaţiului dintre simboluri.

19. Demonul de vacanţă este parte a agentului utilizator sau a agentului de transfer? Desigur, este instalat folosind agentul utilizator, dar cel care trimite replicile este chiar agentul utilizator? Explicaţi răspunsul.

20. POP3 permite utilizatorilor să aducă mesajele de e-mail dintr-o cutie poştală de la distanţă. Aceasta înseamnă că formatul intern al cutiilor poştale trebuie să fie standard pentru ca orice program POP3 de la client să poată să citească cutia poştală de pe orice server de poştă electronică? Discutaţi răspunsul.

21. Din punctul de vedere al unui ISP, POP3 şi IMAP diferă într-o măsură importantă. Utilizatorii POP3 îşi golesc în general cutiile poştale zilnic. Utilizatorii IMAP îşi păstrează mesajele pe server un timp nedefinit. Imaginaţi-vă că vi se cere să sfătuiţi un ISP ce protocol ar trebui să suporte. Ce argumente aţi aduce?

22. Poşta pe Web(Webmail) foloseşte POP3, IMAP sau nici unul? Dacă foloseşte unul din ele, de ce a fost ales acela? Dacă nici unul, care este mai aproape de idee?

23. Când sunt transmise, paginile de Web sunt prefixate de antete MIME. De ce?

24. Când sunt necesare programe de vizualizare externe? Cum ştie un program de navigare pe care să-l folosească?

25. Este posibil ca atunci când un utilizator urmează pe o hiper-legătură în Netscape, să fie pornit un anumit program, iar urmând aceeaşi hiper-legătură în Internet Explorer să fie pornit un program complet diferit, chiar dacă tipul MIME întors în ambele cazuri este identic? Explicaţi răspunsul.

26. Un server Web cu mai multe fire de execuţie este organizat ca în fig. 7-21. Durează 500 iisec să accepte o cerere şi să verifice memoria ascunsă. Jumătate din timp, fişierul este găsit în memoria ascunsă şi este întors imediat. Cealaltă jumătate, modulul trebuie să se blocheze 9 ms până când cererea la disc este adăugată în coadă şi procesată. Câte module ar trebui să aibă serverul pentru a ţine procesorul ocupat tot timpul (presupunând că discul nu reprezintă o gâtuire (bottleneck))?

27. URL-ul standard *http* presupune că serverul de Web ascultă pe portul 80. Totuşi, e posibil ca un server de Web să asculte pe alt port. Născociţi o sintaxă rezonabilă pentru URL pentru accesarea unui fişier pe un port nestandard.

28. Cu toate că nu a fost menţionată în text, o formă alternativă pentru un URL este folosirea adresei IP în loc de numele său DNS. Un exemplu de folosirea a adresei IP este *http://192.31.231.66/index.html*. Cum ştie programul navigator dacă numele ce urmează schema este un nume DNS sau o adresă IP?

29. Imaginaţi-vă că cineva de la Departamentul CS din Stanford a scris un nou program pe care vrea să-l distribuie prin FTP. El pune programul în catalogul *ftp/pub/freebies/newprog.c.* Care este URL-ul probabil pentru acest program?

30. În fig. 7-25, *www.aportal.com* ţine evidenţa preferinţelor utilizatorilor într-un cookie. Un dezavantaj al acestei scheme este că cookie-urile sunt limitate la 4KB şi dacă preferinţele sunt extinse, de exemplu la multe valori ale acţiunilor, echipe sportive, tipuri de ştiri, vremea în multe oraşe, ofertele din diverse categorii de produse şi altele, limita de 4KB ar putea fi insuficientă. Proiectaţi o alternativă pentru păstrarea preferinţelor utilizatorului pentru a nu avea această problemă.

31. Banca Sloth (Trândăvie) doreşte să facă operaţiile bancare mai simple pentru utilizatorii mai leneşi, astfel încât după ce un utilizator se autentifică, banca îi returnează identificatorul de client într-un cookie. Ce părere aveţi de această idee? Va funcţiona? Este o idee buna?

32. În fig. 7-26, în marcajul <IMG> apare parametrul *ALT.* În ce condiţii este folosit de programul de navigare şi cum?

33. Realizaţi o imagine selectabilă în HTML? Daţi un exemplu.

34. Arătaţi cum marcajul <a> poate fi folosit pentru a face şirul „ACM” un hiper-legături către *http://www.acm.org*.

35. Proiectaţi un formular pentru o nouă companie, InterBurger, care permite comanda hamburge- rilor prin Internet. Formularul trebuie să conţină numele clientului, adresa, oraşul, ca şi o opţiune asupra dimensiunii (ori gigant, ori imens) şi o opţiune pentru brânză. Burger-ii urmează a fi plătiţi la livrare cu bani gheaţă, aşa că nu este necesară nici o informaţie despre cartea de credit.

36. Proiectaţi un formular care cere utilizatorului să tasteze două numere. Când utilizatorul apasă pe butonul de trimitere, serverul întoarce suma lor. Scrieţi partea care rulează la server ca un script PHP.

37. Pentru fiecare din aplicaţiile următoare, spuneţi (1) dacă este posibil şi (2) dacă este mai bine să se folosească un script PHP sau JavaScript şi de ce.
(a) Afişarea unui calendar pentru orice lună începând cu septembrie 1752.
(b) Afişarea unui program al zborurilor de la Amsterdam la New York.
(c) Graficul unui polinom cu coeficienţii daţi de utilizator.

38. Scrieţi un program JavaScript care acceptă un întreg mai mare ca 2 şi spune dacă este, sau nu, un număr prim. Notaţi că JavaScript are instrucţiunile if şi while cu aceeaşi sintaxă ca în C sau Java. Operatorul modul este %. Dacă aveţi nevoie de rădăcina pătrată a lui *x,* folosiţi *Math.sqrt(x).*

39. O pagină HTML este de forma:

```
<html> <body>
<a href="www.info-source.com/welcome.html"> Apăsaţi aici pentru informaţii </a>
</body> </html>
```

Dacă utilizatorul apasă pe hiper-legătură, este deschisă o conexiune TCP şi este trimisă o serie de linii la server. Scrieţi toate liniile trimise.

40. Antetul *If-Modified-Since* poate fi folosit pentru a vedea dacă o pagină din memoria ascunsă este încă validă. Cererile pot fi pentru pagini conţinând imagini, sunete, video etc., precum şi HTML. Credeţi că eficacitatea acestei tehnici este mai bună sau mai rea pentru imagini JPEG în comparaţie cu HTML? Gândiţi bine la ceea ce „eficacitate" înseamnă şi explicaţi răspunsul vostru.

41. În ziua unui mare eveniment sportiv, cum ar fi un campionat sportiv, multă lume se duce pe situl Web oficial. Este aceasta o aglomerare bruscă în acelaşi sens cu cel al alegerilor din Florida, în 2000? De ce sau de ce nu?

42. Are sens ca un singur ISP să aibă rolul de CDN? Dacă da, cum ar funcţiona? Dacă nu, care este greşit în legătură cu această idee?

43. În ce condiţii folosirea unui CDN este o idee proastă?

44. Terminalele pentru Web-ul fără fir(Wirelss Web) au o lăţime de bandă mică, ceea ce face importantă o codificare eficientă. Proiectaţi o schemă de transmitere eficientă a textului în engleză pe o legătură fără fir către un dispozitiv WAP. Puteţi presupune că terminalul are câţiva mega- octeţi de memorie ROM şi un procesor moderat de puternic. *Indiciu:* gândiţi-vă cum transmiteţi ceva în japoneză, unde fiecare simbol este un cuvânt.

45. Un CD memorează 650 MB de date. Este folosită compresia pentru CD-uri audio? Explicaţi raţionamentul.

46. În fig. 7-57(c) zgomotul de cuantificare apare datorită folosirii de eşantioane pe 4 biţi pentru a reprezenta nouă valori de semnale. Primul eşantion, la 0, este exact, dar câteva dintre următoarele nu. Care este procentul de eroare la 1/32, 2/32 şi 3/32 din perioadă?

47. Ar putea fi folosit modelul psiho-acustic pentru reducerea lărgimii de bandă necesare pentru telefonia Internet? Dacă da, ce condiţii, dacă există, ar trebui să fie îndeplinite pentru ca el să funcţioneze? Dacă nu, de ce nu?

48. Un server de flux audio are o distanţă pe sens de 50 ms cu un dispozitiv de redare (media plsyer). El emite la 1 Mbps. Dacă media player-ul are o memorie tampon de 1 MB, ce puteţi spune despre poziţia minimă şi cea maximă?

49. Algoritmul de întreţesere din fig. 7-60 are avantajul de a fi capabil să supravieţuiască unei pierderi ocazionale a unui pachet fără a introduce o pauză în redarea sunetului (playback). Totuşi, când este folosit pentru telefonia Internet, el are şi un mic dezavantaj. Care este el?

50. Transmisia de voce-peste-IP are aceleaşi probleme cu zidurile de protecţie (firewalls) ca şi transmiterea fluxurilor audio? Discutaţi răspunsul vostru.

51. Care este rata de biţi pentru transmiterea necomprimată a culorii la 800 x 600

cu 8 biţi/pixel la 40 cadre/sec?

52. Poate o eroare de l-bit într-un cadru MPEG să afecteze mai mult decât cadrul în care a apărut eroarea? Explicaţi răspunsul vostru.

53. Să considerăm exemplul video-serverului cu 100.000 de clienţi, unde fiecare client vizionează două filme pe lună. Jumătate din filme se transmit la 8 seara. Câte filme trebuie să transmită serverul simultan în acest interval de timp? Dacă fiecare film necesită 4 Mbps, câte conexiuni OC-12 necesită serverul pentru reţea?

54. Să presupunem că legea lui Zipf este îndeplinită pentru accese la un server video cu 10.000 de filme. Dacă serverul memorează cele mai populare 1000 de filme pe disc magnetic, iar restul de 9000 pe disc optic, daţi o expresie pentru fracţia tuturor referinţelor care se vor face la discul magnetic. Scrieţi un mic program pentru evaluarea acestei expresii numerice.

55. Unii cyber-băgăcioşi şi-au înregistrat nume de domenii care sunt ortografieri greşite ale siturilor companiilor cunoscute, ca de exemplu, *www.microsfot.com*. Faceţi o listă de cel puţin cinci asemenea domenii.

56. Numeroase persoane au înregistrat nume de domenii DNS de genul *www.cuvant.com*, unde *cuvânt* este un cuvânt obişnuit. Pentru fiecare din categoriile următore, enumeraţi cinci situri Web şi spuneţi pe scurt despre ce este vorba (de exemplu, *www.stomach.com* este un gastroenteorologist din LongIsland). Iată lista de categorii: animale, mâncare, obiecte de gospodărie, părţi ale corpului. Pentru ultima categorie, vă rog rămâneţi la părţile corpului de deasupra taliei.

57. Proiectaţi emoji-uri proprii folosind o hartă de biţi 12 x 12. Includeţi prietenul, prietena, profesorul şi politicianul.

58. Scrieţi un server POP3 care acceptă următoarele comenzi: *USER, PASS, LIST, RETR, DELE* şi *QUIT.*

59. Rescrieţi serverul din fig. 6-6 ca un server Web adevărat, folosind comanda GET de la HTTP 1.1. Ar trebui să accepte şi mesajul *Host.* Serverul trebuie să menţină o memorie ascunsă cu fişierele recent aduse de pe disc şi să servească cererile din această memorie atunci când este posibil.

SECURITATEA REŢELELOR

În primele decenii ale existenţei lor, reţelele de calculatoare au fost folosite de cercetătorii din universităţi pentru trimiterea poştei electronice şi de către funcţionarii corporaţiilor pentru a partaja imprimantele. În aceste condiţii, problema securităţii nu atrăgea prea mult atenţia. Dar acum, când milioane de cetăţeni obişnuiţi folosesc reţelele pentru operaţiuni bancare, cumpărături şi plata taxelor, securitatea reţelei apare la orizont ca o mare problemă potenţială. În acest capitol, vom studia securitatea reţelei din mai multe unghiuri, evidenţiind numeroase pericole şi discutând mulţi algoritmi şi protocoale destinate a face reţelele mai sigure.

Securitatea este un subiect vast şi acoperă o multitudine de imperfecţiuni. În forma sa cea mai simplă, ea asigură că persoane curioase nu pot citi sau, şi mai rău, modifica mesajele adresate altor destinatari. Ea se ocupă de cei care încearcă să apeleze servicii la distanţă, deşi nu sunt autorizaţi să le folosească. De asemenea, securitatea implică verificarea dacă un mesaj, ce pretinde că vine de la IRS şi spune: „Plăteşte până vineri”, provine într-adevăr de la IRS şi nu de la Mafie. Securitatea se ocupă de problemele legate de capturarea şi falsificarea mesajelor autorizate şi de cei ce încearcă să nege faptul că au trimis anumite mesaje.

Majoritatea problemelor de securitate sunt cauzate intenţionat de persoane răuvoitoare care încearcă să obţină anumite beneficii, să atragă atenţia, sau să provoace rău cuiva. Câţiva dintre cei care comit în mod obişnuit astfel de fapte sunt menţionaţi în fig. 8-1. Din această listă trebuie să rezulte clar că realizarea unei reţele sigure implică ceva mai mult decât păstrarea ei fără erori de programare. Aceasta implică surclasarea unor adversari adeseori inteligenţi, dedicaţi şi uneori bine dotaţi material. Trebuie de asemenea să fie clar că măsurile care pot contracara inamici accidentali vor avea un impact redus asupra unor adversari

serioşi. Arhivele poliţiei arată că cele mai multe atacuri nu au fost săvârşite de străini prin ascultarea unor linii telefonice, ci de către angajaţi ranchiunoşi. În consecinţă, sistemele de securitate ar trebui proiectate ţinând seama de acest fapt.

Adversar	Scop
Student	Pentru a se distra furând poşta electronică a celorlalţi
Spărgător	Pentru a testa securitatea sistemului cuiva; pentru a fura date
Responsabil de vânzări	Pentru a pretinde că reprezintă toată Europa, nu numai
Om de afaceri	Pentru a descoperi planul strategic de marketing al
Fost funcţionar	Pentru a se răzbuna că a fost concediat
Contabil	Pentru a sustrage bani de la o companie
Agent de vânzări	Pentru a nega o promisiune făcută clientului prin poştă
Şarlatan	Pentru a fura numere de cărţi de credit şi a le vinde
Spion	Pentru a afla puterea militară a inamicului sau secrete
Terorist	Pentru a fura secrete legate de conflicte armate

Fig. 8-1. Câteva persoane ce generează probleme de securitate şi motivele acestora.

Problemele securităţii reţelei pot fi împărţite, în mare, în patru domenii strâns interconectate: confidenţialitate, autentificare, nerepudiere şi controlul integrităţii. Păstrarea secretului, denumită de asemenea şi confidenţialitate, se referă la păstrarea informaţiei departe de utilizatorii neautorizaţi. Aceasta este ceea ce vine în mintea oamenilor atunci când se gândesc la securitatea reţelei. Autentificarea reprezintă determinarea identităţii persoanei cu care vorbeşti înainte de a dezvălui informaţii importante sau de a intra într-o afacere. Nerepudierea implică semnături: cum să dovedeşti că un client a făcut într-adevăr o comandă pentru zece milioane de nimicuri de 89 de cenţi fiecare, dacă, mai târziu, el pretinde că preţul era de 69 de cenţi? Sau poate susţine că nu a făcut nici o comandă. În fine, cum poţi fi sigur că un mesaj pe care l-ai primit a fost cel trimis cu adevărat şi nu unul pe care un adversar răutăcios l-a modificat în tranzit sau l-a măsluit?

Toate aceste aspecte (confidenţialitate, autentificare, nerepudiere şi controlul integrităţii) apar şi în sistemele tradiţionale, dar cu câteva diferenţe semnificative. Integritatea şi confidenţialitatea sunt realizate prin folosirea poştei înregistrate şi prin sigilarea documentelor. Jefuirea trenului ce duce poşta este mai greu de realizat decât era în zilele lui Jesse James.

De asemenea, oamenii pot de obicei să spună ce diferenţă este între un document original şi o fotocopie şi adeseori numai primul are valoare pentru ei. Ca test, faceţi o fotocopie a unui cec valid. Încercaţi luni să încasaţi de la bancă banii pe cecul original. Apoi încercaţi marţi să încasaţi banii pe fotocopie. Observaţi diferenţa din comportamentul băncii. Cu cecuri electronice, originalul şi copia nu sunt distinctibile. Va trece ceva vreme până când băncile vor şti cum să trateze astfel de situaţii.

Oamenii autentifică alţi oameni prin recunoaşterea feţelor, vocilor şi scrisului lor. Dovada semnării se face prin semnături pe scrisori cu antet, sigilii etc. Falsificarea poate fi de obicei detectată prin scris, hârtie şi experţi în grafologie. Nici una din aceste opţiuni nu este disponibilă electronic. Evident, sunt necesare alte soluţii.

Înainte de a intra în prezentarea acestor soluţii, merită să consumăm câteva minute pentru a stabili unde anume în stiva de protocoale se situează securitatea

reţelei. Probabil că nu există un singur loc. Fiecare nivel poate contribui cu ceva. La nivelul fizic, ascultarea firelor poate fi zădărnicită prin încapsularea liniilor de transmisie în tuburi sigilate conţinând gaz de argon la presiuni înalte. Orice încercare de a perfora tubul va duce la pierderi de gaz, reducând presiunea şi trăgând alarma. Câteva sisteme militare folosesc această tehnică.

La nivelul legătură de date, pachetele transmise pe o linie punct-la-punct pot fi codificate când părăsesc una dintre maşini şi decodificate când intră în cealaltă. Toate detaliile pot fi manipulate la nivelul legătură de date, fără ca nivelurile mai înalte să aibă cunoştinţă de ceea ce se petrece. Această soluţie eşuează, totuşi, atunci când pachetele trebuie să traverseze mai multe rutere, deoarece pachetele trebuie decriptate în fiecare ruter, făcându-le astfel vulnerabile la atacurile din interiorul ruterelor. De asemenea, ea nu permite ca anumite sesiuni să fie protejate (de exemplu, acelea ce implică cumpărăturile on-line folosind cărţi de credit), iar altele nu. Cu toate acestea, **criptarea legăturii** (eng.: link encryption), cum este numită această metodă, poate fi adăugată cu uşurinţă la orice reţea şi este adeseori utilă.

La nivelul reţea, pot fi instalate ziduri de protecţie pentru a păstra pachetele în interior sau pentru a păstra pachetele în afara acestuia. Securitatea IP funcţionează de asemenea la acest nivel.

La nivelul transport, pot fi criptate conexiuni întregi, de la un capăt la celălalt, adică de la un proces la celălalt. Pentru o securitate maximă, este necesară securitatea capăt-la-capăt (eng.: end-to- end security).

În sfârşit, problemele cum sunt autentificarea utilizatorilor şi nerepudierea nu pot fi tratate decât la nivelul aplicaţie.

Din moment ce securitatea nu se potriveşte perfect cu nici un nivel, nu se potriveşte în nici un capitol al acestei cărţi. Din acest motiv, ea are propriul capitol.

Chiar dacă acest capitol este lung, tehnic şi esenţial, el este cvasi-irelevant pentru moment. Este bine ştiut faptul că cele mai multe probleme de securitate la bănci, de exemplu, se datorează angaja- ţilor incompetenţi, procedurilor de securitate slabe, sau fraudelor interne mai degrabă decât unor criminali inteligenţi care ascultă liniile telefonice şi apoi decodează mesajele criptate. Dacă o persoană poate intra într-o filieră oarecare a unei bănci cu o hârtie ATM pe care a găsit-o pe stradă susţinând că a uitat numărul de PIN şi poate primi unul nou pe loc (în numele unor bune relaţii cu clienţii), atunci toată criptografia din lume nu va preveni abuzurile. În această privinţă, cartea lui Roy Anderson vă ajută să „deschideţi ochii”, deoarece documentează sute de exemple de eşecuri ale securităţii în numeroase industrii, aproape toate dintre ele datorându-se unor (ceea ce s-ar putea numi politicos) practici neglijente de afaceri sau lipsei de grijă pentru securitate (Anderson, 2001). Cu toate acestea, noi suntem optimişti că odată cu extinderea comerţului electronic (eng.: ecommerce), companiile îşi vor îmbunătăţi procedurile operaţionale, eliminând această fisură şi aducând din nou aspectele de securitate în centrul atenţiei.

Exceptând securitatea de la nivelul fizic, aproape toată securitatea se bazează pe principii criptografice. Din acest motiv, vom începe studiul nostru asupra securităţii prin examinarea detaliată a criptografiei. În secţiunea 8.1, vom studia câteva dintre principiile de bază. Între secţiunile 8-2 şi 8-5 vom examina câţiva dintre algoritmii şi

structurile de date fundamentale folosite în criptografie. Apoi vom examina în detaliu cum pot fi folosite aceste concepte pentru a obţine securitatea în reţele. Vom concluziona cu câteva scurte gânduri despre tehnologie şi societate.

Înainte de a începe, se impune o ultimă observaţie: ce nu este acoperit. Am încercat să ne concentrăm asupra problemelor din reţea, mai degrabă decât asupra problemelor de sisteme de operare sau aplicaţii, chiar dacă este deseori greu de tras o linie de demarcaţie. De exemplu, nu am spus nimic despre autentificarea utilizatorilor folosind biometria, securitatea parolelor, atacurile prin inundarea tampoanelor de memorie (eng. : buffer overflow), cai Troieni, falsificarea login-ului, bombe logice, viruşi, viermi şi altele. Toate aceste subiecte sunt acoperite în detaliu în Cap. 9 din *Sisteme de operare moderne* (Tanenbaum, 2001). Cititorul interesat de aspectele securităţii sistemelor este îndrumat către această carte. Acum haideţi să începem călătoria noastră.

8.1 CRIPTOGRAFIA

Criptografie provine din cuvintele greceşti pentru „scriere secretă”. Criptografia are o istorie lungă şi pitorească ce datează cu mii de ani în urmă. În această secţiune, vom schiţa doar câteva dintre aspecte, ca informaţii de bază pentru ceea ce urmează. Pentru o istorie completă este recomandată cartea lui Kahn (1995). Pentru o tratare detaliată a situaţiei actuale în securitate şi algoritmi, protocoale şi aplicaţii criptografice, a se vedea (Kaufman et. al 2002). Pentru o abordare mai matematizată, a se vedea (Stinson, 2002). Pentru o abordare mai puţin matematică, a se vedea (Burnett şi Paine, 2001).

Profesioniştii fac o distincţie între cifruri şi coduri. Un **cifru** este o transformare caracter-cu- caracter sau bit-cu-bit a mesajului, fără a ţine cont de structura lingvistică a acestuia. Prin contrast, un **cod** înlocuieşte un cuvânt cu un alt cuvânt sau cu un simbol. Codurile nu mai sunt folosite, deşi au avut o istorie glorioasă. Cel mai reuşit cod inventat vreodată a fost folosit de forţele armate ale S.U.A. în timpul celui de-al doilea război mondial în Pacific. Pur şi simplu au pus indieni Navajo să vorbească între ei folosind cuvinte specifice dialectului Navajo pentru termenii militari, de exemplu *chay-da-gahi-nail-tsaidi* (literal: ucigaş de broaşte ţestoase) pentru armele anti-tanc. Limbajul Navajo este foarte tonal, extrem de complex şi nu are o formă scrisă. Şi nimeni din Japonia nu ştia nimic despre el.

În septembrie 1945, *San Diego Union* descria codul spunând: „Pentru trei ani, oriunde acostau trupele Marine, Japonezii auzeau o mână de zgomote bolborosite răspândite printre alte sunete care semănau cu strigătul unui călugăr Tibetan şi cu sunetul golirii unei sticle cu apă fierbinte. Japonezii nu au spart niciodată codul şi mulţi vorbitori de cod Navajo au fost răsplătiţi cu onoruri militare înalte pentru servicii şi curaj extraordinare. Faptul că S.U.A. a spart codul japonez, dar că japonezii nu au reuşit niciodată să spargă codul Navajo a jucat un rol crucial în victoriile americane din Pacific.

8.1.1 Introducere în criptografie

Din punct de vedere istoric, patru grupuri de oameni au contribuit şi au folosit arta criptografiei: armata, corpurile diplomatice, cei ce au ţinut jurnale şi

îndrăgostiţii. Dintre acestea, armata a avut rolul cel mai important şi a dat contur domeniului de-a lungul secolelor. În interiorul organizaţiilor militare, mesajele ce trebuiau criptate erau de obicei date pentru criptare şi transmitere unor funcţionari codori de nivel scăzut, prost plătiţi. Volumul de mesaje nu permitea ca această muncă să fie făcută doar de câţiva specialişti de elită.

Până la apariţia calculatoarelor, una din marile constrângeri ale criptografiei a fost capacitatea funcţionarilor codori de a realiza transformările necesare, adeseori pe câmpul de luptă, cu echipament redus. O constrângere suplimentară a fost dificultatea de comutare rapidă de la o metodă criptografică la alta, deoarece aceasta implica reantrenarea unui număr mare de oameni. Totuşi, pericolul ca un funcţionar codor să fie capturat de către inamic a făcut să devină esenţială posibilitatea de a schimba metoda criptografică instantaneu, în caz de necesitate. Aceste cerinţe antagoniste au dat naştere modelului din fig. 8-2.

Mesajele ce trebuie criptate, cunoscute sub numele de **text clar** (eng.: plain text), sunt transformate printr-o funcţie parametrizată de o **cheie** (eng.: *key).* Rezultatul procesului de criptare, cunoscut sub numele de **text cifrat** (eng.: ciphertext), este apoi transmis, adeseori prin curier sau radio. Presupunem că inamicul, sau **intrusul**, ascultă şi copiază cu acurateţe tot textul cifrat. Totuşi, spre deosebire de destinatarul la care trebuie să ajungă, el nu ştie care este cheia de decriptare şi astfel nu poate decripta prea uşor textul cifrat. Uneori intrusul poate nu numai să asculte canalul de comunicaţie (intrus pasiv), ci şi să înregistreze mesajele şi să le retransmită mai târziu, să injecteze propriile sale mesaje sau să modifice mesajele legitime înainte ca ele să fi fost preluate de receptor (intrus activ). Arta de a sparge cifruri se numeşte **criptanaliză** (eng.: cryptanalysis). Arta de a sparge cifruri, denumită **criptanaliză** şi arta de a le inventa (criptografia) sunt cunoscute împreună sub numele de **criptologie** (eng.: cryptology).

Text clar Cheie criptare, Text cifrat Cheie decriptare

Fig. 8-2. Modelul de criptare (pentru un cifru cu cheie).

Adesea va fi util să avem o notaţie pentru a pune în relaţie textul clar, textul cifrat şi cheile. Vom folosi $C=E_K(P)$ pentru a simboliza faptul că prin criptarea

textului clar *P,* folosind cheia *K,* rezultă textul cifrat *C*. Similar, $P=D_K(C)$ reprezintă decriptarea lui *C* pentru a obţine din nou textul clar. Rezultă că:

$$D_K(E_K(P))=P$$

Această notaţie sugerează că *E* şi *D* sunt doar nişte funcţii matematice, ceea ce sunt de altfel. Singura parte mai delicată este aceea că ambele sunt funcţii cu doi parametrii iar noi am scris unul dintre parametri (cheia) ca indice, nu ca argument, pentru a face distincţie între ea şi mesaj.

O regulă fundamentală a criptografiei este aceea că trebuie presupusă cunoaşterea de către orice criptanalist a metodelor utilizate pentru criptare şi decriptare. Cu alte cuvinte, criptanalistul ştie cum lucrează în detaliu metoda de criptare *E* şi de decriptare *D* din fig. 8-2. Cantitatea de efort necesară pentru a inventa, testa şi instala o metodă nouă, ori de câte ori vechea metodă este compromisă (sau este presupus a fi compromisă), a făcut întotdeauna nepractică păstrarea secretă a algoritmului de criptare. A crede că este secretă atunci când nu este face mai mult rău decât bine.

Aici intră în scenă cheia. Cheia constă dintr-un şir (relativ) scurt care selectează una dintre mai multe criptări posibile. În contrast cu metoda generală, care poate fi schimbată doar o dată la câţiva ani, cheia poate fi schimbată oricât de des este nevoie. Astfel modelul nostru de bază este stabil şi metoda generală, cunoscută de toată lumea, este parametrizată cu o cheie secretă şi uşor de schimbat. Ideea că algoritmii sunt cunoscuţi de criptanalist şi că secretul constă exclusiv în cheie se numeşte **principiul lui Kerckhoff** denumit astfel după criptograful militar flamand Auguste Kerckhoff care l-a enunţat prima oară în 1883 (Kerckhoff, 1883). Astfel că avem:

Principiul lui Kerckhoff: Toţi algoritmii trebuie să fie publici; numai cheile sunt secrete.

Caracterul nesecret al algoritmului nu poate fi subliniat suficient. Încercarea de a ţine secret algoritmul, cunoscută în domeniu ca **securitate prin obscuritate**, nu funcţionează niciodată. De asemenea, prin publicarea algoritmului, criptograful primeşte consultanţă gratuită de la un număr mare de criptologi din mediul academic, nerăbdători să spargă sistemul pentru a putea publica articole care să demonstreze cât de inteligenţi sunt ei. Dacă mulţi experţi au încercat să spargă algoritmul timp de 5 ani după publicarea lui şi nici unul nu a reuşit, probabil că algoritmul este destul de solid.

Din moment ce adevăratul secret este cheia, lungimea sa reprezintă un aspect foarte important de proiectare. Să considerăm o simplă combinaţie de seif. Principiul general este că se introduc cifre în secvenţă. Oricine ştie aceasta, dar cheia este secretă. O lungime a cheii de două cifre înseamnă că există 100 de posibilităţi. O lungime a cheii de trei cifre înseamnă 1000 de posibilităţi şi o lungime a cheii de şase cifre înseamnă un milion. Cu cât cheia este mai lungă, cu atât este mai mare **volumul de muncă** (eng.: work factor) pe care trebuie să-l depună criptanalistul. Volumul de muncă pentru a sparge sistemul prin căutare exhaustivă în spaţiul cheilor este exponenţial în raport cu lungimea cheii. Secretul provine din a avea un algoritm puternic (dar public) şi o cheie lungă. Pentru a-l

împiedica pe fratele tău mai mic să-ţi citească poşta electronică, sunt suficiente chei de 64 de biţi. Pentru folosirea uzuală în comerţ, trebuie folosiţi cel puţin 128 de biţi. Pentru a păstra la distanţă principalele guverne, sunt necesare chei de cel puţin 256 de biţi, preferabil mai mulţi.

Din punctul de vedere al criptanalistului, problema criptanalizei are trei variaţiuni principale. Când are la dispoziţie o cantitate de text cifrat şi nici un fel de text clar, el este confruntat cu **problema textului cifrat** (eng.: ciphertext only problem). Criptogramele care apar la secţiunea de enigme a ziarelor pun acest tip de problemă. Când criptanalistul are câtva text clar şi textul criptat corespunzător, problema este cunoscută sub numele de **problema textului clar cunoscut** (eng.: known plaintext problem). În sfârşit, atunci când criptanalistul are abilitatea de a cripta bucăţi de text clar la propria sa alegere, avem de-a face cu **problema textului clar ales** (eng.: chosen plaintext problem). Criptogramele din ziare ar putea fi sparte trivial dacă criptanalistului i s-ar permite să pună întrebări de genul: Care este criptarea pentru ABCDEFGHIJKL?

Novicii în domeniul criptografiei presupun adeseori că dacă un cifru poate rezista unui atac de tip text cifrat (eng.: ciphertext only attack), el este sigur. Această presupunere este foarte naivă. În multe cazuri criptanalistul poate ghici corect unele părţi din textul clar. De exemplu, primul lucru pe care multe sisteme îl afişează atunci când sunt accesate este login: . Echipat cu câteva perechi de text clar – text criptat potrivit, sarcina criptanalistului devine mult mai uşoară. Pentru a obţine securitatea, criptograful trebuie să fie prudent şi să se asigure că sistemul este rezistent, chiar dacă inamicul său poate cripta cantităţi arbitrare de text clar ales.

Metodele de criptare au fost istoric împărţite în două categorii: cifruri cu substituţie şi cifruri cu transpoziţie. Vom studia acum pe scurt pe fiecare dintre aceste cifruri, ca informaţie fundamentală pentru înţelegerea criptografiei moderne.

8.1.2 Cifrurile cu substituţie

Într -un **cifru cu substituţie** fiecare literă sau grup de litere este înlocuit pentru a-l deghiza, cu altă literă sau grup de litere. Unul dintre cele mai vechi cifruri cunoscute este **Cifrul lui Caesar**, atribuit lui Julius Caesar. În această metodă, *a* devine *D, b* devine *E, c* devine *F, ..., z* devine *C.* De exemplu, cuvântul *attack* devine *DWWDFN.* În exemple textul clar va fi scris cu litere mici, iar textul cifrat va fi scris cu majuscule.

O uşoară generalizare a cifrului lui Caesar permite alfabetului textului cifrat să fie deplasat cu *k* litere, în loc de a fi deplasat întotdeauna cu 3. În acest caz, *k* devine o cheie pentru metoda generală a alfabetelor deplasate circular. Cifrul Caesar este posibil să îl fi păcălit pe Pompei, dar de atunci el nu mai păcăleşte pe nimeni.

Următoarea îmbunătăţire este de a stabili o corespondenţă pentru fiecare simbol din textul clar, pentru simplitate să spunem cele 26 de litere, cu o altă literă. De exemplu:

textul clar: ab cdefghi jklmnopqrsyuvwxyz
textul cifrat: QWERTYUI OPASDFGHJKLZXCVBNM

Sistemul general de substituire simbol-cu-simbol este denumit **substituţie monoalfabetică** (eng.: monoalphabetic substitution), cheia fiind şirul de 26 de litere corespunzând întregului alfabet. Pentru cheia anterioară, textul clar *attack* va fi transformat în texul cifrat *QZZQEA*.

La prima vedere, acesta ar putea fi considerat un sistem sigur deoarece, deşi criptanalistul cunoaşte sistemul general (substituţie literă cu literă), el nu cunoaşte care dintre cele 26! « 4 x 10^{26} chei posibile este folosită. Spre deosebire de cifrul lui Caesar, încercarea tuturor cheilor nu este o abordare prea promiţătoare. Chiar şi la 1 ns per soluţie, unui calculator i-ar trebui 10^{10} ani pentru a încerca toate cheile.

Totuşi, fiind dată o cantitate surprinzător de mică de text cifrat, cifrul poate fi spart cu uşurinţă. Atacurile de bază folosesc ca informaţie proprietăţile statistice ale limbajelor naturale. În engleză, de exemplu, *e* este cea mai frecvent folosită literă, urmată de *t*, *o*, *a*, *n*, *i* etc. Cele mai comune combinaţii de două litere, sau **digrame** (eng.: *digrams),* sunt *th, în, er, re* şi *an*. Cele mai comune combinaţii de trei litere, sau **trigrame** (eng.: *trigrams*), sunt *the*, *ing*, *and* şi *ion*.

Un criptanalist care încearcă să spargă un cifru monoalfabetic va începe prin a număra frecvenţele relative ale tuturor literelor din textul cifrat. După aceea el trebuie să încerce să asocieze cea mai frecventă literă cu *e*, următoarea cu *t*. Apoi el va căuta trigramele, pentru a o găsi pe cea mai frecventă cu forma *tXe*, care indică cu tărie că *X* este *h*. Similar, dacă apare frecvent şablonul *thYt*, probabil că *Y* îl înlocuieşte pe *a*. Cu această informaţie, el poate căuta aparţiile frecvente ale trigramelor de forma *aZW*, care sunt cel mai probabil, *and*. Prin astfel de presupuneri făcute asupra celor mai comune litere, digrame, trigrame şi cu ceva cunoştinţe despre şabloanele asemănătoare de vocale şi consoane, criptanalistul construieşte literă cu literă o variantă de text clar.

O altă abordare este aceea de a ghici un cuvânt sau o expresie probabilă. De exemplu, consideraţi următorul text cifrat provenind de la o firmă de contabilitate

(împărţit în blocuri de câte cinci caractere):

CTBMN BYCTC BTJDS QXBNS GSTJC BTSWX CTQTZ CQVUI
QJSGS TJQZZ MNQJS VLNSZ VSZJU JDSTS JQUUS JUBXJ
DSKSU JSNTK BGAQJ ZBGYQ TLCTZ BNYBN QJSW

Un cuvânt probabil într-un mesaj provenind de la o firmă de contabilitate este *financial* (rom.: financiar). Folosind propriile noastre cunoştinţe, cum ar fi faptul că *financial* are o literă care se repetă (*i*), cu alte patru litere între apariţiile ei, vom căuta în textul cifrat litere repetate aflate la această distanţă. Găsim 12 potriviri pe poziţiile 6, 15, 27, 31, 42, 48, 56, 66, 70, 71, 76 şi 82. Totuşi, doar două dintre acestea, 31 şi 42 au următoarea literă (cea corespunzând lui *n* în textul clar) repetată în locul corespunzător. Din acestea două, doar 31 are un *a* corect poziţionat, deci ştim că *financial* începe la poziţia 30. Din acest moment, deducerea cheii este simplă folosind statisticile de frecvenţă a literelor în textele englezeşti.

8.1.3 Cifrurile cu transpoziţie

Cifrurile cu substituţie păstrează ordinea simbolurilor din textul clar, dar le deghizează. Spre deosebire de acestea, **cifrurile cu transpoziţie** (eng.: transposition ciphers) reordonează literele, dar nu le deghizează. Fig. 8-3 descrie un cifru cu transpoziţie simplu, transpoziţia pe coloane. Cifrul are drept cheie un cuvânt sau o expresie ce nu conţine litere repetate. În acest exemplu cheia este MEGABUCK. Scopul cheii este să numeroteze coloanele, coloana 1 fiind sub litera din cheie cea mai apropiată de începutul alfabetului ş.a.m.d. Textul clar este scris orizontal, pe rânduri. Textul cifrat este citit pe coloane, începând cu coloana al cărui număr de sub literă este mai mic.

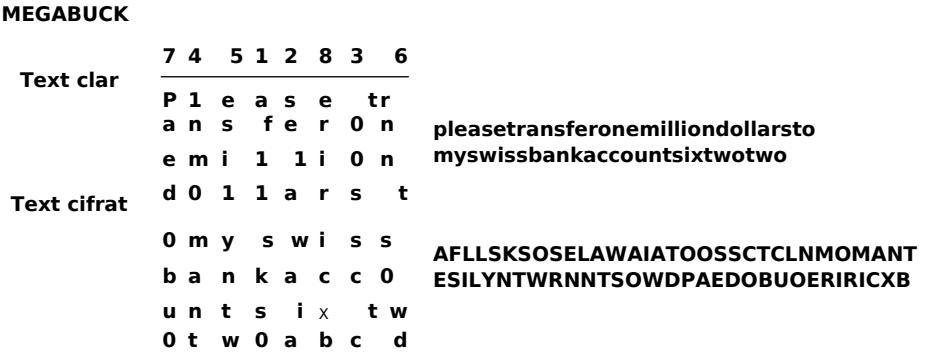

MEGABUCK
7 4 5 1 2 8 3 6
Text clar
P l e a s e t r
a n s f e r 0 n
e m i 1 1 i 0 n
pleasetransferonemilliondollarsto
myswissbankaccountsixtwotwo
Text cifrat
d 0 1 1 a r s t
0 m y s w i s s
b a n k a c c 0
u n t s i x t w
0 t w 0 a b c d
AFLLSKSOSELAWAIATOOSSCTCLNMOMANT
ESILYNTWRNNTSOWDPAEDOBUOERIRICXB

Fig. 8-3. Un cifru cu transpoziţie.

Pentru a sparge un cifru cu transpoziţie, criptanalistul trebuie mai întâi să fie sigur că are de-a face cu un cifru cu transpoziţie. Analizând frecvenţa de apariţie pentru *E*, *T*, *A*, *O*, *I*, *N* etc., este uşor de văzut dacă ele se încadrează în şablonul obişnuit pentru text clar. Dacă da, cifrul este sigur un cifru cu transpoziţie, deoarece într-un astfel de cifru, fiecare literă este reprezentată de ea însăşi, păstrând distribuţia frecvenţelor.

Următorul pas care trebuie făcut este să se emită o presupunere asupra numărului de coloane. În multe cazuri un cuvânt sau o expresie probabilă poate fi ghicită din contextul mesajului. De exemplu, să presupunem că un criptanalist bănuieşte că expresia în text clar *milliondollars* apare pe undeva prin mesaj. Se observă că în urma împachetării acestei expresii, în textul cifrat apar digrame- le *MO, IL, LL, LA, IR* şi *OS.* Litera *O* din textul cifrat urmează după litera *M* din acelaşi text (adică sunt vertical adiacente în coloana 4) deoarece ele sunt separate în expresia probabilă de o distanţă egală cu lungimea cheii. Dacă a fost folosită o

cheie de lungime 7, în locul acestora ar fi trebuit să apară digramele *MD*, *IO*, *LL*, *LL*, *IA*, *OR* şi *NS*. De fapt, pentru fiecare lungime a cheii, în textul cifrat sunt produse seturi diferite de digrame. Prin urmărirea diferitelor posibilităţi, criptanalistul poate determina cu uşurinţă lungimea cheii.

Pasul care mai rămâne este ordonarea coloanelor. Când numărul de coloane, *k,* este mic, poate fi examinată fiecare dintre cele *k(k-1)* perechi de coloane pentru a vedea dacă diagrama de frecvenţe se potriveşte diagramei pentru textul clar în engleză. Perechea cu cea mai bună potrivire se presupune că este corect poziţionată. Acum fiecare coloană ce a rămas este încercată a fi succesorul acestei perechi. Coloana pentru care frecvenţa digramelor şi trigramelor se potriveşte cel mai bine se presupune a fi corectă. Coloana precedentă este găsită în acelaşi mod. Se continuă întregul proces până când o ordine potenţială este descoperită. Există şanse ca textul clar să devină recognoscibil din acest stadiu (de exemplu, dacă apare *milloin,* este clar ce erori există în el).

Anumite cifruri cu transpoziţie acceptă un bloc de lungime fixă la intrare şi produc un bloc de lungime fixă la ieşire. Aceste cifruri pot fi descrise complet dându-se doar o listă în care să se precizeze ordinea în care caracterele vor fi trimise la ieşire. De exemplu, cifrul din fig. 8-3 poate fi văzut ca un cifru bloc pe 64 de caractere. Ieşirea sa este 4, 12, 20, 28, 36, 44, 52, 60, 5, 13, . . . , 62. Cu alte cuvinte, cel de-al patrulea caracter de la intrare, *a,* este primul ce va fi trimis la ieşire, urmat de al doisprezecelea, *f,* ş.a.m.d.

8.1.4 Chei acoperitoare

Construirea unui cifru imposibil de spart este de fapt destul de simplă; tehnica este cunoscută de decenii. În primul rând alegeţi un şir aleatoriu de biţi pe post de cheie. Apoi convertiţi textul clar într-un şir de biţi, de exemplu folosind reprezentarea ASCII. În final, calculaţi XOR (SAU eXclusiv) între cele două şiruri, bit cu bit. Textul cifrat rezultat nu poate fi spart, deoarece pentru un eşantion suficient de mare de text cifrat, fiecare literă va apărea la fel de des, de asemenea şi orice digramă sau trigramă. Această metodă, cunoscută sub numele de **metoda cheilor acoperitoare** (eng.: onetime pad) este imună la toate atacurile din prezent şi din viitor, indiferent de puterea de calcul pe care o are la dispoziţie intrusul. Motivul provine din teoria informaţiei: pur şi simplu nu există nici o informaţie în mesaj deoarece orice text clar posibil cu o lungime dată este la fel de probabil.

Un exemplu de folosire al cheilor acoperitoare este prezentat în fig. 8-4. Mai întâi, mesajul 1, *„ I loveyou.”* (rom.: „Te iubesc”) este convertit în cod ASCII pe 7 biţi. Apoi o cheie acoperitoare, cheia acoperitoare 1, este aleasă şi combinată XOR cu mesajul pentru a obţine textul cifrat. Un criptanalist ar putea să încerce toate cheile acoperitoare posibile pentru a vedea ce text clar rezultă pentru fiecare. De exemplu, cheia acoperitoare 2 din figură poate fi testată, rezultând textul în clar 2, *„Elvis lives”* (rom.: „Elvis trăieşte”) care ar putea să fie sau să nu fie plauzibil (un subiect în afara domeniului acestei cărţi). De fapt, pentru fiecare text clar de 11 caractere ASCII, există o cheie acoperitoare care îl generează. Asta înseamnă ceea vrem să spunem prin afirmaţia că nu există nici o informaţie în mesaj: se poate

obţine din el orice mesaj cu lungimea corespunzătoare.

Mesajul	10010	01000	11011	11011	11101	11001	01000	11110	11011	11101	01011
Cheia	10100	10010	11100	10101	10100	11000	00010	01010	10101	11001	01010
Textul cifrat:	00110 11	11010 11	00111 10	01110 10	01001 00	00001 10	01010 11	10100 11	01110 00	00100 11	00001 01
Cheia acoperitoare 2:	10111 10	00001 11	11010 00	10100 11	10101 11	01001 10	10001 11	01110 10	10011 10	11101 10	11101 10
Textul în	10001	11011	11101	11010	11100	01000	11011	11010	11101	11001	11100

Fig. 8-4. Folosirea unei chei acoperitoare pentru criptare şi posibilitatea de a obţine orice text posibil din textul cifrat prin folosirea altei chei acoperitoare.

Cheile acoperitoare sunt extraordinare în teorie, dar au numeroase dezavantaje în practică. Unul dintre ele este faptul că nu poate fi memorată cheia, astfel încât atât transmiţătorul cât şi receptorul trebuie să poarte cu ei o copie scrisă a acesteia. Dacă vreunul dintre ei este capturat, evident că existenţa cheilor scrise nu este de dorit. În plus, cantitatea totală de date care poate fi transmisă este limitată de dimensiunea cheii disponibile. Dacă spionul dă lovitura şi descoperă o comoară de date, el nu va fi capabil să le transmită înapoi la cartierul general deoarece cheia a fost epuizată. O altă problemă este sensibilitatea metodei la pierderea sau inserarea de caractere. Dacă transmiţătorul şi receptorul pierd la un moment dat sincronizarea, toate datele de aici încolo vor apărea ca fiind eronate.

Odată cu apariţia calculatoarelor, metoda cheilor acoperitoare poate deveni practică pentru anumite aplicaţii. Sursa unei chei poate fi un DVD special care conţine câţiva gigabiţi de informaţie şi care, transportat într-o cutie de DVD pentru filme, precedaţi de câteva minute de imagini, nu ar fi nici măcar suspect. Desigur, la viteza reţelei de ordinul gigabiţilor, a trebui să introduci un nou DVD la fiecare 5 secunde poate deveni obositor. Iar DVD-urile trebuie duse personal de la transmiţător la receptor înainte de transmiterea oricărui mesaj, ceea ce reduce mult valoarea lor practică.

Criptografia cuantică

Interesant este că ar putea exista o soluţie la problema transmiterii cheilor acoperitoare prin reţea, şi ea provine dintr-o sursă neaşteptată: mecanica cuantică. Ideea este încă experimentală, dar testele iniţiale sunt promiţătoare. Dacă poate fi perfecţionată şi eficientizată, teoretic toată criptografia va fi făcută folosind chei acoperitoare dovedit fiind că acestea sunt sigure. În continuare vom explica pe scurt cum funcţionează această metodă, **criptografia cuantică**. În particular, vom descrie un protocol care se numeşte BB84 după autorii săi şi anul publicării (Bennet şi Brassard, 1984).

O utilizatoare, Alice, doreşte să stabilească o cheie acoperitoare cu un al doilea utilizator, Bob. Alice şi Bob sunt denumiţi protagonişti, caracterele principale în povestea noastră. De exemplu, Bob este un bancher cu care Alice ar dori să facă afaceri. Numele „Alice” şi „Bob” au fost folosite pentru protagoniştii principali în aproape fiecare studiu şi lucrare despre criptografie în ultima decadă. Criptografii iubesc tradiţia. Dacă am fi folosit „Andy” şi „Barbara” pentru protagoniştii principali, nimeni nu ar fi crezut nimic din acest capitol. Deci, aşa să fie.

Dacă Alice şi Bob ar putea stabili o cheie acoperitoare, ar putea să o folosească

pentru a comunica sigur. Întrebarea este: Cum pot ei să o stabilească fără să schimbe între ei DVD-uri? Putem presupune că Alice şi Bob sunt la capetele opuse ale unei fibre optice prin care ei pot trimite şi primi pulsuri de lumină. Cu toate acestea, o intrusă agitată, Trudy, poate tăia fibra pentru a introduce un dispozitiv de ascultare activ. Trudy poate citi toţi biţii din ambele direcţii. Ea poate de asemenea să trimită mesaje false în ambele direcţii. Situaţia ar părea fără speranţă pentru Alice şi Bob, dar criptografia cuantică poate pune subiectul într-o nouă lumină.

Criptografia cuantică este bazată pe faptul că lumina circulă în mici pachete numite **fotoni**, care au nişte proprietăţi specifice. În plus, lumina poate fi polarizată prin trecerea ei printr-un filtru pola- rizator, un fapt bine cunoscut, atât de purtătorii de ochelari de soare cât şi de fotografi. Dacă o rază de lumină (adică un flux de fotoni) trece printr-un filtru polarizator, toţi fotonii care ies din el vor fi polarizaţi pe direcţia axului filtrului (adică pe verticală). Dacă raza va fi trecută acum printr-un al doilea filtru polarizator, intensitatea luminii care iese din al doilea filtru va fi proporţională cu rădăcina cosinusului unghiului dintre cele două axe. Dacă axele sunt perpendiculare, nu va trece nici un foton. Nu contează orientarea absolută ale celor două filtre; contează doar unghiul dintre axele lor.

Pentru a genera o cheie acoperitoare, Alice are nevoie de două seturi de filtre polarizatoare. Primul set constă dintr-un filtru vertical şi un filtru orizontal. Această alegere se numeşte **bază recti- liniară**. O bază (la plural: baze) este doar un sistem de coordonate. Al doilea set de filtre este similar, cu excepţia faptului că sunt rotite cu 45 de grade, astfel că un filtru este de la stânga jos la dreapta sus, iar celălalt filtru este de la stânga sus la dreapta jos. Această alegere se numeşte **bază diagonală**. Deci Alice are două baze, pe care le poate introduce în raza de lumină oricând doreşte. În realitate, Alice nu are patru filtre separate, ci un cristal a cărui polarizare poate fi comutată electric cu o viteză foarte mare la oricare dintre cele patru direcţii permise. Bob are acelaşi echipament ca şi Alice. Faptul că Alice şi Bob au fiecare câte două baze disponibile este esenţial în criptografia cuantică.

Pentru fiecare bază, Alice desemnează o direcţie ca 0 şi cealaltă ca 1. În exemplul prezentat mai jos, vom presupune că ea alege direcţia verticală ca 0 şi cea orizontală ca fiind 1. Independent, ea alege de asemenea diagonala stânga-jos dreapta-sus ca 0 şi stânga-sus dreapta-jos ca 1. Ea trimite aceste opţiuni lui Bob în text necriptat.

Acum Alice alege o cheie acoperitoare, bazată de exemplu pe un generator de numere aleatoare (un subiect foarte complex în sine). Ea o transferă bit cu bit lui Bob, alegând aleator una dintre cele două baze disponibile pentru fiecare bit. Pentru a trimite un bit, tunul de fotoni emite un foton polarizat corespunzător pentru baza pe care o foloseşte pentru bitul respectiv. De exemplu, ea ar putea să aleagă bazele diagonală, rectiliniară, rectiliniară, diagonală, rectiliniară, etc. Pentru a trimite cheia sa acoperitoare 1001110010100110 cu aceste baze, ea va trimite fotonii din fig. 8-5(a). Fiind date cheia acoperitoare şi secvenţa bazelor, polarizarea folosită pentru fiecare bit este unic determinată. Biţii trimişi câte un foton la un moment dat se numesc **qubiţi.**

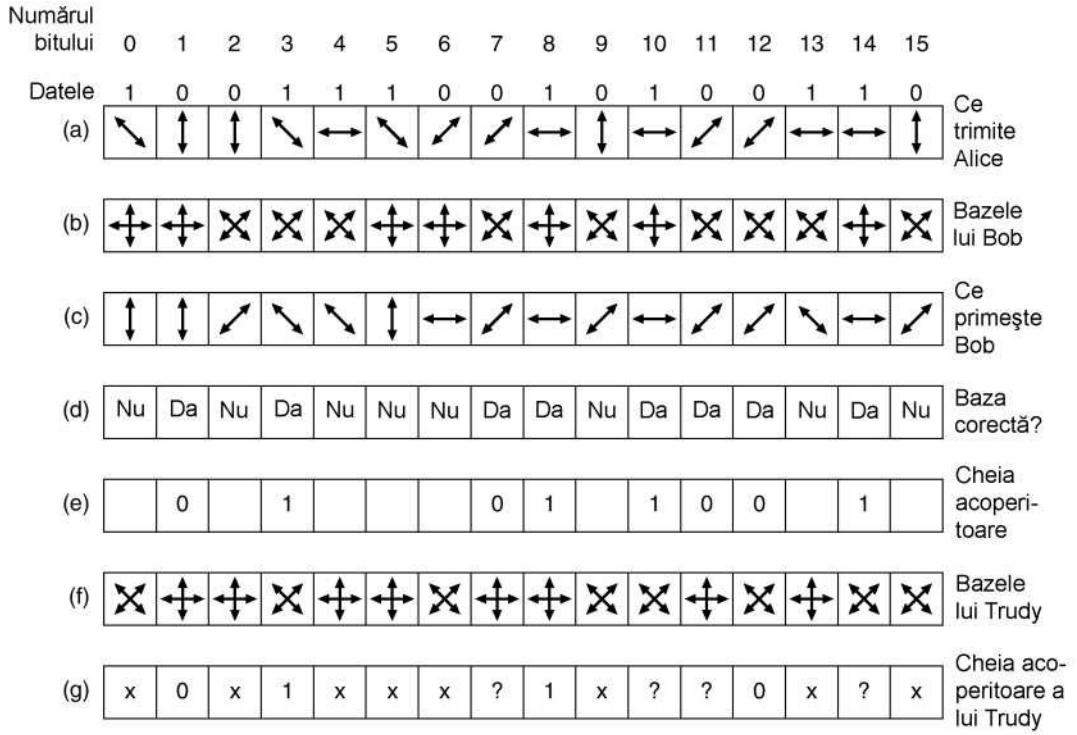

Numărul bitului
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Datele
1 0 0 1 1 1 0 0 1 0 1 0 0 1 1 0
(a)
Ce trimite Alice
(b)
Bazele lui Bob
(c)
Ce primeşte Bob
(d)
Nu Da Nu Da Nu Nu Nu Da Da Nu Da Da Da Nu Da Nu
Baza corectă?
(e)
0 1 0 0 1 0 0 1
Cheia acoperitoare
(f)
Bazele lui Trudy
(g)
x 0 x 1 x x x ? 1 x ? ? 0 x ? x
Cheia acoperitoare a lui Trudy

Fig. 8-5. Un exemplu de criptografie cuantică

Bob nu ştie ce baze să folosească, prin urmare pentru fiecare foton sosit, alege o bază aleatoare şi o foloseşte, după cum se poate vedea în fig. 8-5(b). Dacă alege baza corectă, el va obţine bitul corect. Dacă alege baza incorectă, el obţine un bit aleator deoarece dacă un foton trece printr-un filtru polarizat la 45 de grade faţă de polarizarea proprie, sare aleator la polarizarea filtrului sau sare cu o probabilitate egală, la o polarizare perpendiculară cu cea a filtrului. Această proprietate a fotonilor este fundamentală pentru mecanica cuantică. Prin urmare, câţiva dintre biţi vor fi corecţi iar câţiva vor fi aleatori, dar Bob nu ştie care sunt unii şi care sunt ceilalţi. Rezultatele lui Bob sunt prezentate în fig. 8-5(c).

Dar cum află Bob ce baze a ghicit corect şi pe care le-a greşit? Pur şi simplu, el îi va spune lui Alice în text clar ce bază a folosit pentru fiecare bit, iar ea îi va spune care sunt corecte şi care sunt greşite tot în clar, după cum este arătat în fig. 8-5(d). Din această informaţie, fiecare dintre ei poate construi un şir de biţi din presupunerile corecte, după cum arată fig. 8-5(e). În medie, acest şir va fi jumătate din lungimea şirului iniţial de biţi, dar deoarece ambele părţi îl cunosc, îl pot folosi pe post de cheie acoperitoare. Alice trebuie să trimită un şir puţin mai mare decât dublul lungimii dorite pentru ca ea şi Bob să obţină o cheie acoperitoare cu lungimea dorită. Problemă rezolvată!

Dar staţi un pic! Am uitat de Trudy. Să presupunem că ea este curioasă să ştie ce are Alice de spus şi taie fibra, inserând propriul ei detector şi transmiţător. Din nefericire pentru ea, nici ea nu ştie ce bază să folosească pentru fiecare foton. Tot ce poate să facă este să aleagă aleatoriu o bază pentru fiecare foton, cum a făcut şi Bob. Un exemplu cu alegerile ei este prezentat în fig. 8-5(f). Atunci când Bob

raportează mai târziu (în text clar) ce baze a folosit şi Alice îi spune (în text clar) care dintre ele sunt corecte, Trudy va şti ce a înţeles bine şi ce nu a înţeles bine. În fig. 8-5 ea a nimerit bine biţii 0, 1, 2, 3, 4, 6, 8, 12 şi 13. Dar ea ştie din replica lui Alice că doar biţii 1, 3, 7, 8, 10, 11, 12 şi 14 fac parte din cheia acoperitoare. Pentru patru dintre aceşti biţi (1, 3, 8, 12) ea a ghicit corect şi a capturat bitul corect . Ea nu a ghicit ceilalţi patru (7, 10, 11 şi 14) biţi şi nu cunoaşte bitul transmis. Prin urmare, Bob ştie că cheia acoperitoare începe cu 01011001, din fig. 8-5(e) dar tot ce are Trudy este 01?1??0?, din fig.8-5(g).

Bineînţeles că Alice şi Bob sunt conştienţi că Trudy ar fi putut captura o parte din cheia lor acoperitoare, astfel că ar dori să minimizeze informaţia deţinută de Trudy. Ei pot face acest lucru prin efectuarea unei transformări asupra acestei chei. De exemplu, ei ar putea diviza cheia acoperitoare în blocuri de câte 1024 biţi, ar putea ridica la pătrat fiecare bloc pentru a forma un număr de 2048 de biţi şi ar putea folosi concatenarea acestor numere de 2048 de biţi ca cheie acoperitoare. Având cunoştinţe parţiale despre şirul de biţi transmis, Trudy nu are nici un mod de a genera pătratul lui şi deci nu dispune de nici o informaţie. Transformarea din cheia acoperitoare originală într-una diferită care reduce cunoştinţele lui Trudy se numeşte **amplificarea confidenţialităţii**. În practică, în locul ridicării la pătrat sunt folosite transformări complexe în care fiecare bit de ieşire depinde de fiecare bit de intrare.

Biata Trudy. Nu numai că nu are nici o idee despre cheia acoperitoare, dar nici prezenţa ei nu este un secret. Ea trebuie să redirecteze fiecare bit recepţionat de la Alice pentru Bob pentru a-l păcăli şi a-l face să creadă că vorbeşte cu Alice. Problema este că tot ce poate ea să facă este să trimită qubitul pe care l-a recepţionat, folosind polarizarea pe care a folosit-o pentru a-l recepţiona şi în jumătate din cazuri ea va greşi, provocând multe erori în cheia acoperitoare a lui Bob.

Când în final Alice va începe să trimită datele, ea le va codifica folosind un puternic cod preventiv corector de erori. Din punctul lui Bob de vedere, o eroare de 1 bit în cheia acoperitoare este echivalentă cu o eroare de transmisie de 1 bit. În ambele cazuri, el va obţine un bit eronat. Dacă există suficient cod preventiv corector de erori, el va putea recupera mesajul original în pofida tuturor erorilor, dar va putea foarte uşor să numere câte erori au fost corectate. Dacă acest număr este cu mult mai mare decât rata prevăzută de erori a echipamentului, el va şti că Trudy a interceptat linia şi va putea să acţioneze în consecinţă (de ex., să-i spună lui Alice să comute pe un canal radio, să cheme poliţia, etc.). Dacă Trudy ar dispune de o metodă de a clona un foton pentru a avea un foton pe care să-l inspecteze şi un foton identic pe care să-l trimită lui Bob, ea ar putea evita detecţia, dar în prezent nu este cunoscută nici o metodă de a clona perfect un foton. Dar chiar dacă Trudy ar putea clona fotoni, asta nu ar reduce valoarea criptografiei cuantice în stabilirea cheilor acoperitoare.

Deşi a fost demonstrat că criptografia cuantică poate opera pe distanţe de 60 km de fibră, echipamentul este complex şi scump. Totuşi, ideea este promiţătoare. Pentru mai multe informaţii despre criptografia cuantică, a se vedea (Mullins, 2002).

8.1.5 Două principii criptografice fundamentale

Deşi în paginile ce urmează vom studia diferite sisteme criptografice, pentru toate acestea există două principii de bază a căror înţelegere este importantă.

Redundanţa

Primul principiu este acela că toate mesajele criptate trebuie să conţină redundanţă, adică informaţie ce nu este necesară pentru înţelegerea mesajului. Un exemplu poate clarifica de ce este nevoie de aceasta. Să considerăm o companie ce se ocupă cu comenzile prin poştă The Couch Potato (TCP), cu 60000 de produse. Crezând că vor fi foarte eficienţi, programatorii de la TCP au decis că mesajele de comandă trebuie să conţină un nume de client pe 16 octeţi, urmat de un câmp de date pe 3 octeţi (1 octet pentru cantitate şi 2 octeţi pentru numărul produsului). Ultimii 3 octeţi vor fi criptaţi folosind o cheie foarte lungă, cunoscută doar de client şi de TCP.

La prima vedere sistemul pare sigur şi, într-un anumit sens, chiar este, deoarece intruşii pasivi nu pot decripta mesajele. Din nefericire, există o slăbiciune fatală a acestui sistem, care îl face inutilizabil. Să presupunem că o funcţionară recent concediată vrea să se răzbune pe TCP pentru că a dat-o afară. Chiar înainte de a pleca, ea ia lista clienţilor cu ea. Ea lucrează în timpul nopţii şi scrie un program care generează comenzi fictive folosind nume de clienţi reali. Deoarece nu posedă lista cheilor, ea pune numere aleatorii în ultimii 3 octeţi şi trimite sute de comenzi la TCP.

Când sosesc aceste mesaje, calculatorul TCP foloseşte numele clientului pentru a localiza cheia şi a decripta mesajul. Din nefericire pentru TCP, aproape fiecare mesaj de 3 octeţi este valid, iar calculatorul începe să tipărească instrucţiunile trimise. Deşi pare ciudat ca un client să comande 837 de seturi de leagăne pentru copii sau 540 de cutii cu nisip, calculatorul poate crede că acesta plănuieşte să deschidă o mulţime de locuri de joacă. În acest mod, un intrus activ (ex-funcţionara) poate cauza probleme imense, chiar dacă ea nu poate înţelege mesajele pe care le generează calculatorul ei.

Problema poate fi rezolvată prin adăugarea unor informaţii redundante tuturor mesajelor. De exemplu, dacă mesajele de comandă sunt extinse la 12 octeţi, dintre care primii 9 trebuie să fie zerouri, atunci acest atac nu ar mai fi funcţional, deoarece ex-funcţionara nu mai poate genera un şir mare de mesaje valide. Morala povestirii este aceea că toate mesajele trebuie să conţină o cantitate considerabilă de informaţie redundantă, astfel încât intruşii activi să nu poată trimite mesaje aleatorii care să fie interpretate ca mesaje valide.

Totuşi, adăugarea informaţiei redundante facilitează spargerea mesajelor de către criptanalişti. Să presupunem că afacerea de comenzi prin poştă este foarte competitivă şi competitorul principal al companiei The Coach Potato, The Sofa Tuber, ar vrea tare mult să ştie câte cutii de nisip vinde TCP. În consecinţă, ei ascultă linia telefonică a TCP. În schema originală, cu mesaje de 3 octeţi, criptanaliza era aproape imposibilă, deoarece după ghicirea unei chei, criptanalistul nu avea cum să-şi dea seama dacă a ghicit corect. În fond, aproape orice mesaj este tehnic corect. Cu noua schemă de 12 octeţi, este uşor pentru criptanalist să distingă un mesaj valid de unul invalid. Astfel că avem:

Principiul criptografic 1: Mesajele trebuie să conţină redundanţă.

Cu alte cuvinte, după decriptarea unui mesaj, receptorul trebuie să poată distinge dacă este valid printr-o simplă inspectarea a acestuia şi prin execuţia unui calcul simplu. Această redundanţă este necesară pentru a împiedica intruşii activi să înşele receptorul trimiţându-i un mesaj fals şi determinându-l să acţioneze în numele mesajului decriptat. Cu toate acestea, aceeaşi redundanţă facilitează intruşilor pasivi spargerea sistemului, deci aici apar unele probleme. Mai mult decât atât, redundanţa nu trebuie niciodată să fie folosită sub forma a *n* zerouri la începutul sau sfârşitul unui mesaj, deoarece trecerea unor astfel de mesaje prin anumiţi algoritmi criptografici dă rezultate pre- dictibile, simplificând criptanaliza. Un cod CRC polinomial ar fi o alegere mult mai bună decât un şir de 0 deoarece receptorul îl poate verifica uşor, dar pentru criptanalist el reprezintă o muncă în plus. Şi mai bună ar fi folosirea unei dispersii criptografice (eng.: cryptographic hash), un concept pe care-l vom explora mai târziu.

Să ne întoarcem un moment la criptografia cuantică şi să vedem care este rolul redundanţei acolo. Datorită interceptării fotonilor de către Trudy, câţiva dintre biţii cheii acoperitoare a lui Bob vor fi eronaţi. Bob are nevoie de redundanţă în mesajele primite pentru a determina faptul că există erori. O formă foarte primitivă de redundanţă este transmiterea mesajului de două ori. Dacă cele două copii nu sunt identice, Bob ştie că fie fibra optică are foarte multe zgomote, fie cineva interfe- rează cu transmisia. Bineînţeles că trimiterea de două ori a fiecărui lucru este extrem de ineficient: o metodă mult mai eficientă de a detecta şi corecta erorile este folosirea unui cod Hamming sau Reed-Solomon. Dar trebuie să fie clar că o anumită redundanţă este necesară pentru a putea distinge între un mesaj valid şi unul invalid, mai ales în prezenţa unui intrus activ.

Prospeţimea

Cel de-al doilea principiu criptografic este acela că trebuie luate anumite măsuri pentru ne asigura că fiecare mesaj primit poate fi verificat că este proaspăt, adică a fost trimis foarte recent. Această măsură este necesară pentru a împiedica intruşii activi să retransmită mesaje mai vechi. Dacă nu se iau nici un fel de astfel de măsuri, ex-funcţionara noastră ar putea asculta linia telefonică a TCP şi ar putea retransmite mesajele valide trimise anterior. Reformulând această idee obţinem:

Principiul criptografic 2: Este necesară o metodă pentru a dejuca atacurile prin replicarea mesajelor

O astfel de măsură este de a include în fiecare mesaj o amprentă de timp validă doar pentru, să spunem, 10 secunde. Receptorul trebuie doar să păstreze mesajele primite în ultimele 10 secunde, pentru a compara mesajele nou sosite cu anterioarele şi pentru a filtra duplicatele. Mesajele mai vechi decât 10 secunde pot fi aruncate, deoarece orice replică a lor trimisă mai târziu de 10 secunde va fi refuzată ca fiind prea veche. Alte măsuri în afara amprentelor de timp vor fi discutate mai târziu.

8.2 ALGORITMI CU CHEIE SECRETĂ

Criptografia modernă utilizează aceleaşi idei de bază ca şi criptografia tradiţională (transpoziţia şi substituţia) dar accentul este diferit. Tradiţional, criptografii foloseau algoritmi simpli. În zilele
noastre este adevărat contrariul: obiectivul este de a face algoritmii de criptare atât de complecşi şi ireversibili, încât, chiar dacă un criptanalist achiziţionează cantităţi imense de text cifrat la alegerea sa, el nu poate face nimic cu el fără a avea cheia.

Prima clasă de algoritmi de criptare pe care o vom studia în acest capitol se numesc **algoritmi cu cheie secretă** (eng.: symmetric-key algorithms) pentru că folosesc aceeaşi cheie pentru criptare şi decriptare. Fig. 8-2 ilustrează folosirea unui algoritm cu cheie secretă. În particular, ne vom concentra asupra **cifrurilor bloc**, care primesc la intrare un bloc de n biţi de text clar şi îl transformă folosind cheia într-un bloc de text cifrat de n biţi.

Algoritmii criptografici pot fi implementaţi fie în hardware (pentru viteză) sau în software (pentru flexibilitate). Deşi cea mai mare parte a acestei lucrări tratează algoritmii şi protocoalele, care sunt independente de implementarea efectivă, ar putea fi interesante câteva cuvinte despre construirea de hardware pentru criptare. Transpoziţiile şi substituţiile pot fi implementate cu circuite simple. Fig. 8-6(a) prezintă un dispozitiv, cunoscut sub numele de **cutie P** (P vine de la permutare), folosit pentru a efectua o transpoziţie asupra unei intrări de 8 biţi. Dacă cei 8 biţi sunt selectaţi să fie notaţi de sus în jos cu 01234567, ieşirea acestei cutii P particulare este 36071245. Printr-o cablare internă corespunzătoare, o cutie P poate fi construită să efectueze orice transpoziţie şi să o facă practic cu viteza luminii.

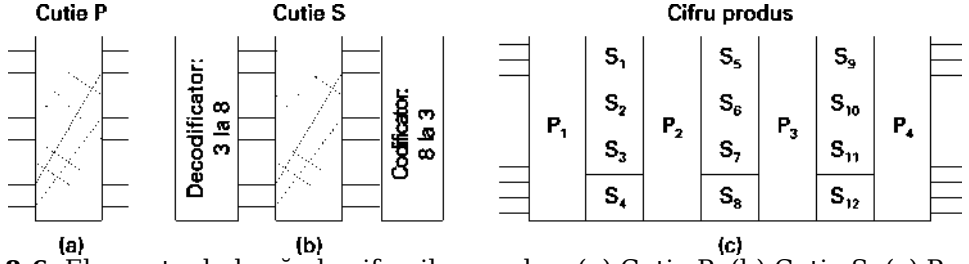


Fig. 8-6. Elemente de bază ale cifrurilor produs. (a) Cutie P. (b) Cutie S. (c) Produs.

Substituţiile sunt realizate de **cutiile S**, după cum este arătat în fig. 8-6(b). În acest exemplu este introdus un text clar de 3 biţi, iar la ieşire este furnizat un text cifrat pe 3 biţi. Intrarea de 3 biţi selectează una dintre cele opt linii ce ies din primul nivel şi o poziţionează pe 1; toate celelalte linii sunt 0. Cel de-al doilea nivel este o cutie P. Ce de-al treilea nivel codifică din nou în binar linia selectată la intrare. Cu cablajul arătat, dacă opt numere scrise în octal 01234567 ar fi fost introduse unul după celălalt, secvenţa de ieşire ar fi 24506713. Cu alte cuvinte, 0 a fost înlocuit cu 2, 1 a fost înlocuit cu 4 etc. Din nou, prin cablarea corespunzătoare a cutiei P în interiorul cutiei S, poate fi realizată orice substituţie. De altfel, un astfel de dispozitiv poate fi construit în hardware şi poate atinge viteze foarte mari deoarece codificatorul şi decodificatorul au doar una sau două întârzieri de porţi

logice (sub o nanosecundă) iar timpul de propagare prin cutia P poate fi mult mai mic decât o picosecundă.

Puterea efectivă a acestor elemente de bază devine vizibilă doar atunci când conectăm în cascadă o serie întreagă de cutii pentru a forma un **cifru produs**, după cum este arătat în fig. 8-6(c). În acest exemplu, 12 linii de intrare au fost transpuse (adică permutate) de primul nivel. Teoretic, ar fi posibil să avem la doilea nivel o cutie S care să pună în corespondenţă un număr de 12 biţi cu alt număr de 12 biţi. Totuşi, un astfel de dispozitiv ar necesita 2^{12}=4096 cabluri încrucişate la nivelul său din mijloc. În schimb, intrarea este împărţită în patru grupuri de 3 biţi, fiecare fiind substituit independent de celelalte. Cu toate că această metodă este mai puţin generală, ea este încă puternică. Prin inclu

derea unui număr suficient de mare de niveluri în cifrul produs, ieşirea poate deveni o funcţie extrem de complicată depinzând de intrare.

Cifrurile produs care operează asupra intrărilor de k biţi pentru a produce ieşiri de k biţi sunt destul de obişnuite. O valoare tipică pentru k este de la 64 la 256. O implementare hardware are de obicei cel puţin 18 niveluri fizice, nu doar şapte ca în fig. 8-6(c). O implementare software este programată ca o buclă cu cel puţin 8 iteraţii, fiecare dintre ele executând substituţii ca cele ale cutiilor S pe sub-blocuri din blocurile de date cu dimensiuni de la 64 la 256 de biţi, urmate o permutare care combină ieşirile cutiilor S. De obicei există o permutare iniţială specială şi, de asemenea, una la început. În literatură, iteraţiile se numesc **runde**.

8.2.1 DES – Data Encryption Standard

În ianuarie 1977, guvernul SUA a adoptat ca standard oficial pentru informaţiile nesecrete un cifru produs şi dezvoltat de IBM. Acest cifru, **DES** (eng.: **Data Encryption Standard –** rom.: Standard pentru Criptarea Datelor), a fost adoptat extensiv în industrie pentru a fi utilizat în produsele de securitate. El nu mai este de mult sigur în forma sa originală, dar într-o formă modificată el este încă util. Vom explica acum cum lucrează DES.

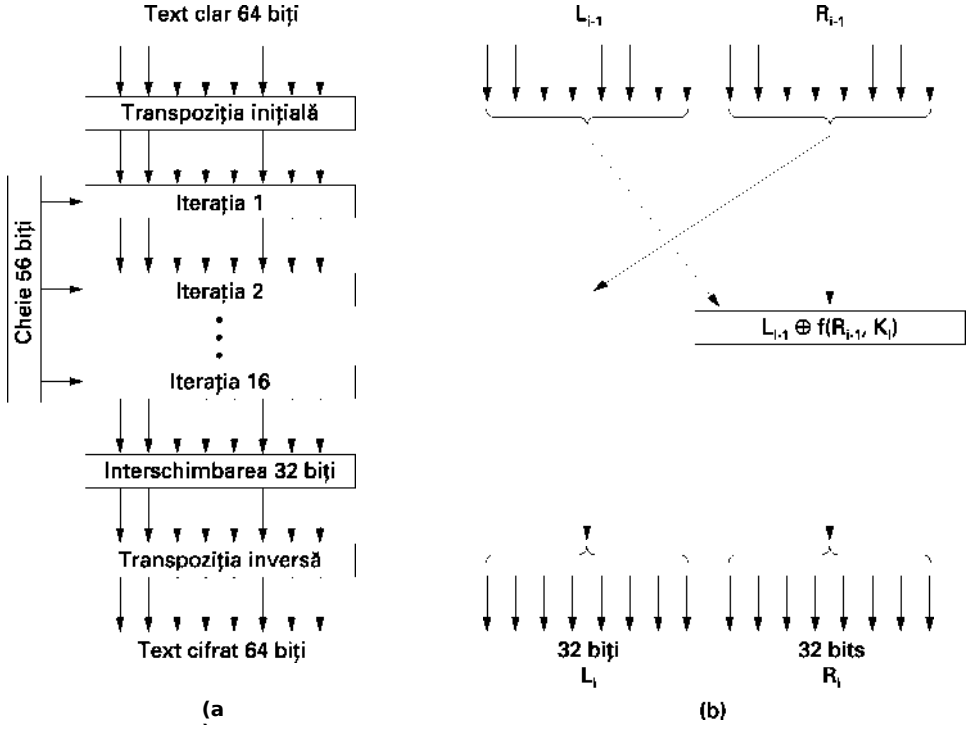


Fig. 8-7. Cifrul DES. (a) Schemă generală. (b) Detalierea unei iteraţii. Semnul + înconjurat de un cerc înseamnă SAU exclusiv

O prezentare generală a DES este făcută în fig. 8-7(a). Textul clar este criptat în blocuri de câte 64 de biţi, rezultând blocuri de 64 de biţi de text cifrat. Algoritmul, care este parametrizat cu o cheie de 56 de biţi, are 19 runde distincte. Prima rundă este o transpoziţie independentă de cheie, aplicată asupra textului clar de 64 de biţi. Ultima rundă este exact inversa acestei transpoziţii. Penultima rundă schimbă cei mai din stânga 32 de biţi cu cei mai din dreapta 32 de biţi. Cele 16 runde rămase sunt funcţional identice dar sunt parametrizate de funcţii de cheie diferite. Algoritmul a fost proiectat pentru a permite ca decriptarea să se facă cu aceeaşi cheie ca şi criptarea, o proprietate necesară în orice algoritm cu cheie secretă. Paşii sunt parcurşi în ordine inversă.

Funcţionarea unuia dintre paşii intermediari este ilustrată în fig. 8-7(b). Fiecare rundă ia două intrări de 32 de biţi şi produce două ieşiri de 32 de biţi. Ieşirea din stânga este o simplă copie a intrării din dreapta. Ieşirea din dreapta rezultă în urma unui SAU exclusiv (XOR) bit cu bit între intrarea din stânga şi o funcţie depinzând de intrarea din dreapta şi de o cheie pentru această rundă, *K*. Toată complexitatea rezidă în această funcţie.

Funcţia constă din patru paşi, parcurşi în secvenţă. În primul rând, este construit un număr de 48 de biţi, *E,* prin expandarea celor 32 de biţi ai lui R_{i-1} în concordanţă cu o regulă de transpoziţie fixă şi de duplicare. În al doilea rând, *E* şi *K* sunt combinate prin XOR. Ieşirea este apoi împărţită în opt grupuri de câte 6 biţi şi fiecare dintre acestea este introdus într-o cutie S diferită. Fiecare dintre cele 64 de intrări posibile într-o cutie S este pusă în corespondenţă cu o ieşire de 4 biţi. În final, aceşti 8 x 4 biţi sunt trecuţi printr-o cutie P.

În fiecare din cele 16 iteraţii este folosită o cheie diferită. Înainte de începerea algoritmului este aplicată o transpoziţie de 56 de biţi asupra cheii. Chiar înainte de începerea fiecărei iteraţii, cheia este partiţionată în două unităţi de câte 28 de biţi, fiecare dintre ele este rotită la stânga cu un număr de biţi depinzând de numărul iteraţiei. *K* este derivat din această cheie rotită prin aplicarea unei transpoziţii pe 56 de biţi asupra ei. La fiecare rundă este extrasă şi permutată o altă submulţime de 48 de biţi din cei 56 de biţi.

O tehnică folosită uneori pentru a face DES mai puternic se numeşte **albire** (eng.: whitening). Ea consistă în efectuarea operaţiei SAU exclusiv între o cheie aleatoare de 64 de biţi cu fiecare bloc de text clar înainte de a-l introduce în DES şi apoi efectuarea încă a unui SAU exclusiv cu o a doua cheie de 64 de biţi a textului cifrat înainte de a-l transmite. Albirea poate fi eliminată simplu prin efectuare operaţiilor inverse (dacă receptorul are cele 2 chei de albire). Din moment ce această tehnică adaugă mai mulţi biţi la lungimea cheii, face căutarea exhaustivă a spaţiului cheii mult mai mare consumatoare de timp. Se poate observa că aceeaşi cheie de albire este folosită pentru fiecare bloc (adică există o singură cheie de albire).

DES a fost învăluit în controverse încă din ziua în care a fost lansat. El se baza pe un cifru dezvoltat şi brevetat de IBM, numit Lucifer, cu excepţia faptului că cifrul IBM-ului folosea o cheie de 128 de biţi în locul uneia de 56 de biţi. Atunci când guvernul federal al SUA a dorit să standardizeze un cifru pentru folosire nesecretă, el a „invitat” IBM-ul să „discute” această problemă cu NSA, agenţia spărgătoare de coduri a guvernului, care are ca angajaţi cel mai mare număr de matematicieni şi

criptologi din lume. NSA este atât de secretă, încât în industrie a apărut următoarea glumă:

Î: Ce înseamnă NSA?
R: No Such Agency (Nu există o astfel de agenţie).

De fapt, NSA provine de la National Security Agency (rom.: Agenţia Naţională de Securitate).

După ce au avut loc aceste discuţii, IBM a redus cheia de la 128 de biţi la 56 de biţi şi a decis să păstreze secret procesul prin care a fost proiectat DES-ul. Mulţi oameni suspectează faptul că lungimea cheii a fost redusă pentru a exista siguranţa că NSA poate sparge DES-ul, dar nici o organizaţie cu un buget mai mic nu ar putea să o facă. Păstrarea secretului proiectării a fost făcută probabil pentru a ascunde o trapă (uşă ascunsă) care ar putea uşura spargerea DES-ului de către NSA. Când un angajat al NSA a atenţionat discret IEEE să abandoneze o conferinţa de criptografie planificată, acesta nu i-a făcut pe oameni să se simtă mai bine. NSA a negat totul.

În 1977, doi cercetători în criptografie de la Stanford, Diffie şi Hellman (1977), au proiectat o maşină pentru a sparge DES-ul şi s-a estimat că ea poate fi construită cu un buget de 20 de milioane de dolari. Dată fiind o mică bucată de text clar şi textul cifrat corespunzător, această maşină ar putea să găsească cheia, prin căutarea exhaustivă a 2^{56} intrări din spaţiul cheilor, în mai puţin de o zi. În prezent, o astfel de maşină ar costa mult sub 1 milion de dolari.

Triplu DES

Încă din 1979, IBM a realizat că lungimea cheii DES era prea mică şi a conceput un mod de a o creşte efectiv, folosind tripla criptare (Tuchman, 1979). Metoda aleasă, care de atunci a fost încorporată în Standardul Internaţional 8732, este ilustrată în fig. 8-8. Aici sunt folosite două chei şi trei runde. În prima rundă textul clar este în mod obişnuit criptat cu K_1. În a doua rundă, este rulat DES în mod de decriptare, folosind cheia K_2. În final, este efectuată o altă criptare cu K_1.

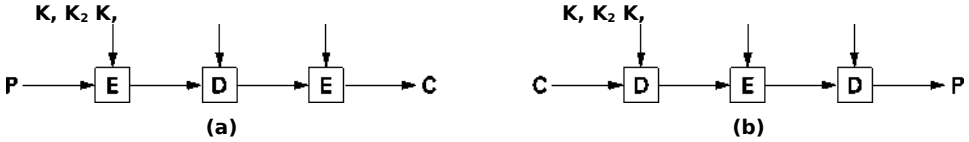


Fig. 8-8. (a)Tripla criptare folosind DES. (b) Decriptarea.

Această proiectarea dă naştere imediat la două întrebări. Prima, de ce sunt folosite doar două chei în loc de trei? A doua, de ce este folosită succesiunea de transformări **EDE** (eng.: **Encrypt Decrypt Encrypt**, rom.: Criptare Decriptare Criptare), în loc de **EEE** (eng.: **Encrypt Encrypt Encrypt**, rom.: Criptare Criptare Criptare)? Motivul pentru care sunt utilizate două chei este acela că chiar şi cei mai paranoici criptografi admit că 112 biţi sunt suficienţi pentru aplicaţiile comerciale de rutină, la momentul actual. A extinde la 168 de biţi înseamnă a adăuga o supraîncărcare inutilă pentru gestiunea şi transportul unei alte chei cu un câştig real redus.

Motivul pentru criptare, decriptare şi apoi iar criptare este compatibilitatea cu produsele existente ce folosesc sisteme DES cu o singură cheie. Atât funcţia de

criptare cât şi cea de decriptare sunt corespondenţe între mulţimi de numere pe 64 de biţi. Din punct de vedere criptografic, cele două corespondenţe sunt la fel de puternice. Totuşi, folosind EDE în locul EEE, un calculator ce utilizează tripla criptare poate comunica cu unul ce foloseşte criptarea simplă, folosind $K_1=K_2$. Această proprietate permite triplei criptări să fie pusă în practică treptat, lucru care nu interesează pe criptografii din mediul academic, dar care este de o importanţă considerabilă pentru IBM şi clienţii săi.

8.2.2 AES – Advanced Encryption Standard

Deoarece DES a început să se apropie de sfârşitul utilei sale vieţi , chiar şi cu DES triplu, **NIST** (**National Institute of Standards and Technology**, rom: Institutul Naţional de Standarde şi Tehnologie), agenţia Departamentului de Comerţ al Statelor Unite însărcinată cu aprobarea standardelor pentru Guvernul Federal al S.U.A., a decis că guvernul are nevoie de un nou standard criptografic pentru folosinţă publică. NIST avea cunoştinţă de controversa din jurul DES şi ştia bine că dacă ar fi anunţat un nou standard orice persoană care ştia câte ceva despre criptografie ar fi presupus în mod automat că NSA a construit o uşă secretă prin care NSA să poată citi orice criptat cu DES. În aceste condiţii, probabil nimeni n-ar fi folosit standardul şi acesta ar fi murit probabil singur.

De aceea NIST a avut o abordare surprinzător de diferită pentru o birocraţie guvernamentală: a sponsorizat un concurs de criptografie. În ianuarie 1997, cercetători din toată lumea au fost invitaţi să depună propuneri pentru un nou standard, care urma să se numească **AES** (**Advanced Encryption Standard**, rom: Standard de Criptare Avansat). Regulile concursului erau:

1. Algoritmul trebuie să fie un cifru bloc simetric.
2. Tot proiectul trebuie sa fie public
3. Trebuie să fie suportate chei de 128, 192, şi de 256 biţi
4. Trebuie să fie posibile atât implementări hardware cât şi software
5. Algoritmul trebuie să fie public sau oferit cu licenţă nediscriminatorie.

Au fost făcute cincisprezece propuneri serioase şi au fost organizate conferinţe publice în care propunerile au fost prezentate, iar publicul a fost încurajat să găsească punctele slabe în fiecare dintre ele. În august 1998, NIST a selectat cinci finalişti, în principal pe criterii de securitate, eficienţă, flexibilitate şi cerinţe de memorie (importante pentru sistemele integrate). Au avut loc mai multe conferinţe şi s-au mai eliminat din variante. La ultima conferinţă s-a organizat un vot liber. Finaliştii şi scorurile lor au fost următoarele:

1. Rijndael (din partea lui Joan Daemen şi Vincent Rijmen, 86 voturi)
2. Serpent (din partea lui Ross Anderson, Eli Biham şi Lars Knudsen, 59 voturi)
3. Twofish (din partea unei echipe condusă de Bruce Schneier, 31 voturi)
4. RC6 (din partea RSA Laboratories, 23 voturi)
5. MARS (din partea IBM, 13 voturi)

În octombrie 2000, NIST a anunţat că votează şi el pentru Rijndael, iar în noiembrie 2001 Rijndael a devenit standard guvernamental al S.U.A. publicat ca Standard Federal de Procesare a Informaţiilor nr.197 (Federal Information

Processing Standard FIPS 197). Datorită deschiderii extraordinare a competiţiei, a proprietăţilor tehnice ale Rijndael şi a faptului că echipa câştigătoare consta din doi tineri belgieni (despre care cu greu se poate presupune că au realizat şi o uşă secretă doar pentru a face pe plac NSA), se aşteaptă ca Rijndael să devină standardul criptografic dominant în lume pentru cel puţin o decadă. Numele Rijndael, pronunţat Rhine-doll (mai mult sau mai puţin), e derivat din numele autorilor: Rijmen + Daemen.

Rijndael permite lungimi de chei şi mărimi de blocuri de la 128 de biţi la 256 de biţi în paşi de câte 32 biţi. Lungimea cheii şi lungimea blocului pot fi alese în mod independent. Cu toate acestea, AES specifică faptul că mărimea blocului trebuie să fie de 128 de biţi şi lungimea cheii trebuie să fie de 128, 192, sau 256 de biţi. E îndoielnic că cineva va folosi vreodată cheile de 192 de biţi, astfel că de fapt, AES are două variante: bloc de 128 de biţi cu cheie de 128 de biţi şi bloc de 128 de biţi cu cheie de 256 de biţi.

În prezentarea algoritmului, vom examina doar cazul 128/128 pentru că acesta va deveni cel mai probabil standardul comercial. O cheie de 128 de biţi permite un spaţiu al cheilor de 2 ® 2 x 10 chei. Chiar dacă NSA reuşeşte să construiască o maşină cu un miliard de procesoare, fiecare fiind capabil să evalueze o cheie în fiecare picosecundă, ar trebui pentru o astfel de maşină aproximativ

10^{10} ani pentru a căuta în spaţiul de chei. Până atunci soarele s-ar stinge, astfel că cei de atunci vor trebui să citească rezultatele la lumina lumânării.

Rijndael

Dintr-o perspectivă matematică, Rijndael se bazează pe Teoria Câmpului Galois, care îi conferă o serie de proprietăţi de securitate demonstrabile. Cu toate acestea, poate fi privit şi ca un cod C, fără a intra în explicaţii matematice.

Ca şi DES, Rijndael foloseşte substituţie şi permutări, ca şi runde multiple. Numărul de runde depinde de mărimea cheii şi de mărimea blocului, fiind 10 pentru o cheie de 128 de biţi cu blocuri de 128 biţi şi mărindu-se până la 14 pentru cheia cu cea mai mare dimensiune şi blocul cu cea mai mare dimensiune. Totuşi, spre deosebire de DES, toate operaţiile sunt la nivel de octet, pentru a permite implementări eficiente hardware şi software. Codul este dat în fig. 8-9.

```
#define LENGTH 16                                /* # octeţi în blocul de date sau în cheie */
#define NROWS 4                                   /* număr   de linii din stare */
#define NCOLS 4                                   /* număr   de coloane din stare */
#define ROUNDS 10                                 /* număr   de iteraţii */
typedef unsigned char byte;                       /* întreg   fără semn pe 8 biţi */
rijndael(byte plaintext[LENGTH], byte ciphertext[LENGTH], byte key[LENGTH]) {
int r;                                           /* index pentru iterare */
byte state[NROWS][NCOLS];                        /* starea curentă */
struct {byte k[NROWS][NCOLS];} rk[ROUNDS + 1]; /* cheile pentru runde */

expand_key(key, rk);                              /* construieşte cheile pentru runde */
copy_plaintext_to_state(state, plaintext);        /* iniţializează starea curentă */
xor_roundkey_into_state(state, rk[0]);            /* face XOR între cheie şi stare */

for (r = 1; r <= ROUNDS; r++)
   { substitute(stare);                           /* aplică substituţia fiecărui octet */
   rotate_rows(state);                            /* roteşte rândul i cu i octeţi */
   if (r < ROUNDS) mix_columns(state);            /* funcţie de amestecare */
   xor_roundkey_into_state(state, rk[r]);         /* face XOR între cheie şi stare */
}
copy_state_to_ciphertext(ciphertext, state); }    /* întoarce rezultatul */
```

Fig. 8-9. Rijndael în linii generale

Funcţia rijndael are trei parametri. Aceştia sunt: plaintext, un vector de 16 octeţi conţinând datele de intrare, ciphertext, un vector de 16 octeţi în care va fi introdus rezultatul cifrat, şi key, cheia de 16 octeţi. Pe durata calculelor, starea curentă a datelor e păstrată într-un vector de octeţi, state, a cărui mărime este NROWS x NCOLS. Pentru blocuri de 128 de octeţi, acest vector este de 4 x 4 octeţi. Întregul bloc de date de 128 de biţi poate fi stocat în 16 octeţi.

Vectorul state este iniţializat cu textul clar şi este modificat la fiecare pas al calculului. În anumiţi paşi, este realizată substituţia octet-cu-octet. În alţii, octeţii sunt permutaţi în interiorul vectorului. Sunt folosite şi alte transformări. La sfârşit, conţinutul lui state reprezintă textul cifrat.

Codul începe prin expandarea cheii în 11 vectori de aceeaşi lungime cu starea. Aceştia sunt memoraţi în rk, care este un vector de structuri, fiecare structură

conţinând un vector stare. Unul dintre aceştia va fi folosit la începutul calculului şi ceilalţi 10 vor fi folosiţi în timpul celor 10 runde, câte unul în fiecare rundă. Calculul cheilor de runde din cheia de criptare este prea complicat pentru a fi prezentat aici. Este de ajuns să spunem că cheile de runde sunt produse prin rotiri repetate şi aplicarea de operaţii XOR asupra unor grupuri de biţi din cheie. Pentru toate detaliile, a se vedea (Daemen şi Rijmen, 2002).

Următorul pas este acela de a copia textul clar în vectorul state astfel încât să poată fi procesat pe perioada rundelor. Acesta este copiat în ordinea coloanelor, cu primii patru octeţi în coloana 0, următorii patru octeţi în coloana 1 şi aşa mai departe. Atât coloanele cât şi liniile sunt numerotate pornind de la 0, deşi rundele sunt numerotate pornind de la 1. Setarea iniţială a celor 12 vectori de octeţi de dimensiuni 4 x 4 este ilustrată în fig. 8-10.

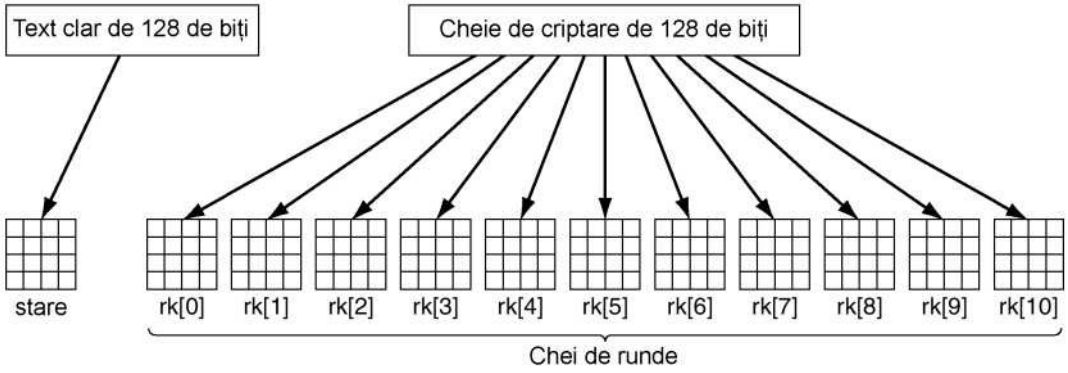


Fig. 8-10. Crearea vectorilor *state* şi *rk*

Mai este un pas înainte de a începe calculul principal: rk[0] este combinat prin XOR în state, octet cu octet. Cu alte cuvinte fiecare octet din cei 16 aflaţi în state este înlocuit cu rezultatul aplicării operaţiei XOR asupra sa şi asupra octetului corespunzător din rk[0].

Şi acum urmează partea cea mai interesantă. Bucla execută 10 iteraţii, câte una pe rundă, transformând state la fiecare iteraţie. Fiecare rundă constă în patru paşi. Pasul 1 realizează o substituţie octet-cu-octet asupra lui state. Pe rând, fiecare octet este folosit ca index într-o cutie S pentru a-i înlocui valoarea prin conţinutul corespunzător din acea cutie S. Acest pas este un cifru de substituţie monoalfabetică directă. Spre deosebire de DES, care are mai multe cutii S, Rijndaek are doar o cutie S.

Pasul 2 roteşte la stânga fiecare din cele 4 rânduri. Rândul 0 este rotit cu 0 octeţi (nu e schimbat), rândul 1 este rotit cu 1 octet, rândul 2 este rotit cu 2 octeţi şi rândul 3 este rotit cu 3 octeţi. Acest pas difuzează conţinutul datelor curente în jurul blocului, analog cu permutările din fig. 8-6.

Pasul 3 amestecă fiecare coloană independent de celelalte. Această amestecare este realizată prin înmulţire de matrice, în care noua coloană este produsul dintre vechea coloană şi o matrice constantă, multiplicarea fiind realizată folosind câmpul finit Galois, GF(2^8). Deşi acest lucru poate părea complicat, există un algoritm care permite fiecărui element al noii coloane să fie calculat folosind două căutări în tabele şi trei operaţii XOR (Daemen şi Rijmen, 2002, Anexa E).

În fine, pasul 4 aplică operaţia XOR pentru cheia din runda curentă şi vectorul stare.

Deoarece fiecare pas e reversibil, decriptarea poate fi realizată prin rularea algoritmului de la coadă la cap. Oricum, este posibilă şi o schemă prin care decriptarea poate fi realizată prin rularea algoritmului de criptare nemodificat, dar folosind tabele diferite.

Algoritmul a fost proiectat nu doar pentru o securitate foarte solidă, dar şi pentru o viteză foarte mare. O bună implementare software pe o maşină la 2 GHz ar trebui să atingă o rată de criptare de 700 Mbps, ceea ce este suficient de rapid pentru a criptarea peste 100 de fişiere video MPEG-2 în timp real. Implementările hardware sunt chiar mai rapide.

8.2.3 Moduri de cifrare

În ciuda acestei complexităţi, AES (sau DES sau orice cifru bloc de altfel) este de fapt un cifru de substituţie monoalfabetică care foloseşte caractere „mari” (caractere de 128 de biţi pentru AES şi caractere de 64 de biţi pentru DES). Ori de câte ori acelaşi bloc de text clar intră pe la un capăt, acelaşi text cifrat iese pe la celălalt. Dacă criptezi textul clar *abcdefgh* de 100 de ori cu aceeaşi cheie DES, obţii acelaşi text cifrat de 100 de ori. Un intrus poate să exploateze această proprietate pentru a submina cifrul.

Modul cu carte de coduri electronică

Pentru a vedea cum poate fi folosită această proprietate a cifrului cu substituţie monoalfabetică pentru a submina DES-ul, vom folosi (triplu) DES deoarece este mai uşor să prezinţi blocuri de 64 de biţi decât blocuri de 128 de biţi; AES are însă exact aceeaşi problemă. Modul direct de a folosi DES pentru a cripta o bucată lungă de text clar este de a o sparge în blocuri consecutive de 8 octeţi (64 de biţi) şi de a le cripta unul după altul cu aceeaşi cheie. Ultima bucată de text clar este completată până la 64 de biţi dacă este nevoie. Tehnica este cunoscută ca modul ECB (Electronic Code Book, rom: modul cu carte de coduri electronică), analog cu cărţile de coduri electronice de modă veche unde era precizat fiecare cuvânt de text clar, urmat de textul său cifrat (de obicei un număr zecimal de 5 cifre).

În fig. 8-11 este prezentat începutul unui fişier conţinând primele anuale ale unei companii care s-a decis să-şi premieze angajaţii. Fişierul constă din înregistrări de 32 de octeţi, câte o înregistrare pentru fiecare angajat, în formatul arătat: 16 octeţi pentru nume, 8 octeţi pentru funcţia ocupată şi 8 octeţi pentru primă. Fiecare din cele 16 blocuri de 8 octeţi (numerotate de la 0 la 15) este criptat cu (triplu) DES.

Nume		Funcţie	Bonus
A|d|a|m|S|,| |L	e|S|1 1 i |e| 1 1	C|1e1 r|k| 1 1	$1 1 1 1 1 1 1 1
B| 1 |a|C|k| , 1 |R	01b1'lnl 1 1 1	B|°|S|S| 1 1 1	$ 1 5 1 0 1 0 1 , 1 0 1 0 1 0
C|°| 1 1 1 1 i |n|S| ,	lKl'lml 1 1 1	Mla1n1a191e1 r1	$|1|0|0|,|0|0|0
Dlalvl'Is!'1 lB	olblbl'Ie! 1 1	J 1 a 1 n 1 ' 1 1 1 0 1 r 1	$l I I I I I I5

Octeţi ----------16----------8----------► -*----------8----------►

Fig. 8-11. Textul clar al unui fişier criptat ca 16 blocuri DES

Leslie a avut o controversă cu şeful şi nu aşteaptă prea mult de la această primă. În schimb Kim este favorita şefului şi oricine ştie asta. Leslie poate avea acces la fişier după ce el a fost criptat, dar înainte de a fi trimis la bancă. Poate Leslie să rectifice această situaţie nedreaptă, dat fiind doar fişierul criptat?

Nici o problemă. Tot ceea ce are Leslie de făcut este să realizeze o copie a celui de-al 12-lea bloc de text cifrat (care conţine prima lui Kim) şi să-l folosească pentru a înlocui blocul de text cifrat cu numărul 4 (care conţine prima lui Leslie). Chiar şi fără a şti ce cuprinde blocul 12, Leslie se poate aştepta să aibă un Crăciun mai fericit anul acesta. (Copierea blocului de text cifrat 8 este de asemenea o posibilitate, dar este mai probabil să fie descoperită; în plus, Leslie nu este o persoană lacomă).

Modul cu înlănţuirea blocurilor cifrate

Pentru a para acest tip de atac, toate cifrurile bloc trebuie înlănţuite în diferite moduri astfel încât înlocuirea unui bloc în modul în care a făcut-o Leslie să conducă la situaţia în care textul clar decriptat, începând cu blocul înlocuit, să fie nefolositor. Un mod de înlănţuire este înlănţuirea blocurilor cifrate (eng.: cipher block chaining). În această metodă, prezentată în fig. 8-12, fiecare bloc de text clar este combinat prin XOR cu blocul anterior de text cifrat, înainte de a fi criptat. În consecinţă, acelaşi bloc de text clar nu se va mai pune în corespondenţă cu acelaşi bloc de text cifrat, iar criptarea nu mai este o mare substituţie monoalfabetică. Primul bloc este combinat prin XOR cu un vector de iniţializare, IV (Initialization Vector), ales aleatoriu, care este transmis împreună cu textul cifrat.

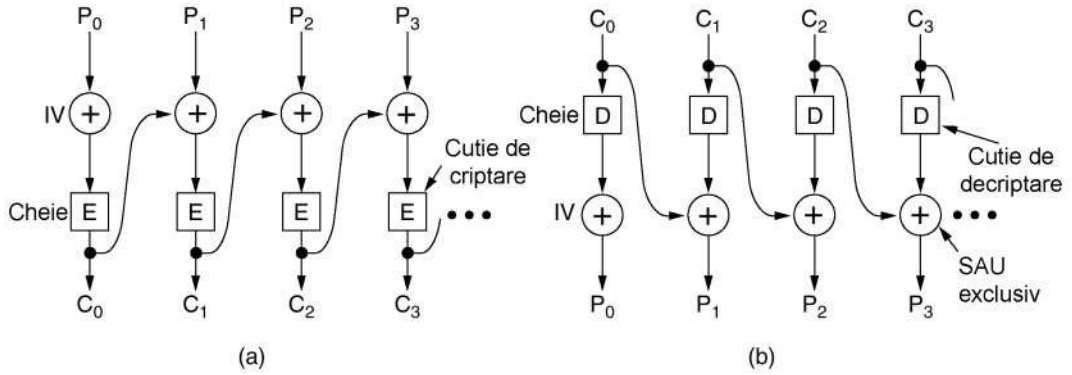


Fig. 8-12. Înlănţuirea blocurilor cifrate (Cipher block chaining). (a) Criptare. (b) Decriptare.

Putem vedea cum lucrează înlănţuirea blocurilor cifrate prin examinarea exemplului din fig. 812. Putem începe prin a calcula $C_0=E(P_0 \text{XOR } IV)$. Apoi vom calcula $C_1= E(P_i \text{ XOR } C_0)$ şi aşa mai departe. Decriptarea foloseşte tot XOR pentru a inversa procesul, cu $P_0=IV \text{XOR } D(C_0)$, şi aşa mai departe. De notat că criptarea blocului *i* este o funcţie de toate textele clare din blocurile de la 0 la *i*- 1, astfel încât acelaşi text clar va genera text cifrat diferit în funcţie de locul unde apare. O

transformare de tipul celei făcute de Leslie va avea ca rezultat un nonsens în cele două blocuri ce încep din câmpul de primă al lui Leslie. Pentru un ofiţer de securitate perspicace, această caracteristică poate sugera de la cine să pornească investigaţia. Înlănţuirea blocurilor cifrate are de asemenea avantajul că acelaşi bloc de text clar nu va rezulta în acelaşi bloc de text cifrat, făcând criptanaliza mai dificilă. De fapt, acesta este principalul motiv pentru care este folosită.

Modul cu reacţie cifrată

Cu toate acestea, înlănţuirea blocurilor cifrate are dezavantajul că este necesar ca un întreg bloc de 64 de biţi să sosească înainte ca decriptarea să poată începe. Acest mod este nepotrivit pentru folosirea în cazul terminalelor interactive, unde o persoană poate să introducă linii mai scurte de 8 caractere şi apoi să se oprească în aşteptarea unui răspuns. Pentru criptările octet-cu-octet poate fi utilizat **modul cu reacţie cifrată** (eng.:**Cipher feedback mode**), folosind (triplu) DES, ca în fig. 8-13. Pentru AES ideea este aceeaşi, numai că se foloseşte un registru de deplasare de 128 de biţi. În această figură, este arătată starea maşinii de criptare după ce octeţii 0 până la 9 au fost criptaţi şi trimişi. Când soseşte blocul 10 din textul clar, după cum este ilustrat în fig. 8-13(a), algoritmul DES operează asupra registrului de deplasare 64 de biţi pentru a genera 64 de biţi de text cifrat. Octetul cel mai din stânga al textului cifrat este extras şi combinat prin XOR cu P_{10}. Acest octet este transmis pe linie. În plus, registrul de deplasare este deplasat cu 8 biţi la stânga, provocând ieşirea lui C_2 pe la capătul din stânga şi inserarea lui C_{10} în poziţia care tocmai a rămas vacantă la dreapta lui C_9. De notat că conţinutul registrului de deplasare depinde de întreaga istorie anterioară a textului clar, astfel încât un şablon care se repetă de mai multe ori în textul clar va fi criptat de fiecare dată diferit în textul cifrat. La fel ca la înlănţuirea blocurilor cifrate, este necesar un vector de iniţializare pentru a porni rostogolirea mingii.

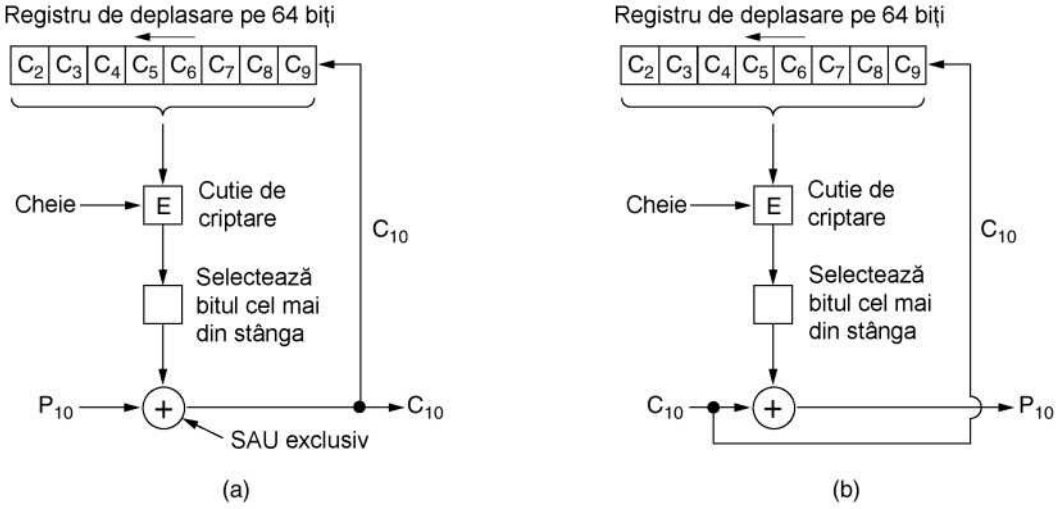


Fig. 8-13. Modul cu reacţie cifrată (Cipher feedback mode). (a) Criptare. (b) Decriptare.

Decriptarea în modul cu reacţie cifrată face acelaşi lucru ca şi criptarea. În

particular, conţinutul registrului de deplasare este *criptat,* nu *decriptat,* astfel încât octetul selectat care este combinat prin XOR cu C_{10} pentru a obţine P_{10} este acelaşi cu cel ce a fost combinat prin XOR cu P_{10} pentru a-l obţine pe C_{10} prima dată. Atâta vreme cât cele două registre de deplasare rămân identice, decriptarea lucrează corect. Acest lucru este ilustrat în fig. 8-13(b).

O problemă a modului cu reacţie cifrată este că, dacă un bit al textului cifrat este inversat accidental în timpul transmisiei, cei 8 octeţi care sunt decriptaţi în timp ce octetul greşit este în registrul de deplasare vor fi corupţi. Odată ce octetul greşit este împins afară din registrul de deplasare, se va genera din nou text clar corect. Astfel, efectele unui singur bit inversat sunt relativ localizate şi nu distrug restul mesajului, dar distrug atâţia biţi câţi are ca lăţime registrul de deplasare.

Modul cu cifru înlănţuit

Cu toate acestea, există aplicaţii în care a avea o eroare de transmisie de 1 bit care să strice 64 de biţi de text clar reprezintă o pierdere prea mare. Pentru aceste aplicaţii există o a patra opţiune, **modul cu cifru înlănţuit** (eng.: **Stream Cipher Mode**). Acesta lucrează criptând un vector de iniţializare, folosind o cheie pentru a obţine un bloc de ieşire. Blocul de ieşire este apoi criptat folosind cheia pentru a obţine un al doilea bloc de ieşire. Acest bloc este apoi criptat pentru a obţine un al treilea bloc şi aşa mai departe. Secvenţa (arbitrar de mare) de blocuri de ieşire, numită **lanţul de chei** (eng.: **keystream**), este tratată ca cheie acoperitoare (eng.: one-time pad) şi este combinată cu textul clar pentru a obţine textul cifrat, ca în fig. 8-14(a). De notat că vectorul IV este folosit doar la primul pas. După aceasta, ieşirea este criptată. De notat de asemenea că lanţul de chei este independent de date, deci poate fi calculat în avans, dacă este nevoie, şi este complet insensibil la erori de transmisie. Decriptarea este arătată în fig. 8-14(b).

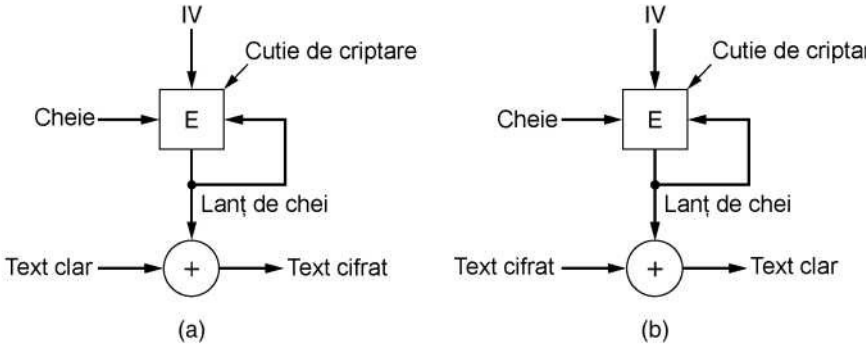


Fig. 8-14. Un cifru înlănţuit. (a) Criptare. (b) Decriptare.

Decriptarea se face generând acelaşi lanţ de chei la receptor. Deoarece lanţul de chei depinde doar de IV şi de cheie, el nu este afectat de erori de transmisie în textul cifrat. Astfel, o eroare de 1 bit în textul cifrat transmis generează doar o eroare de 1 bit în textul clar decriptat.

Este esenţial să nu se folosească aceeaşi pereche (cheie, IV) de două ori cu un cifru înlănţuit deoarece acest lucru va genera acelaşi lanţ de chei de fiecare dată. Folosirea aceluiaşi lanţ de chei de două ori expune textul cifrat la un **atac prin refolosirea lanţului de chei** (eng.: **keystream reuse attack**). Imaginaţi-vă că blocul

de text clar, P_0 este criptat cu lanţul de chei pentru a obţine P_0 XOR K_0. Mai târziu, un al doilea bloc de text clar Q_0 este criptat cu acelaşi lanţ de chei pentru a obţine Q_0 XOR K_0. Un intrus care capturează ambele texte cifrate poate face simplu XOR între ele pentru a obţine P_0 XOR Q_0, ceea ce elimin ă cheia. Intrusul are acum combinaţia XOR a celor două blocuri de text clar. Dacă unul dintre ele este ştiut sau poate fi ghicit, atunci şi celălalt poate fi găsit. În orice caz, un XOR a două şiruri de text clar poate fi atacat folosind proprietăţile statistice ale mesajului. De exemplu, pentru text în limba engleză, cel mai comun caracter din şir va fi probabil XOR între două spaţii, urmat de XOR între spaţiu şi litera “e”. Pe scurt, echipat cu un XOR între două texte clare, criptanalistul are o şansă excelentă de a le deduce pe ambele.

Modul contor

O problemă pe care o au toate modurile în afară de modul cu carte de coduri electronică este că e imposibilitatea accesului aleator la datele cifrate. De exemplu, să presupunem că un fişier este transmis printr-o reţea şi apoi este stocat pe disc în forma criptată. Acesta ar fi un procedeu rezonabil când calculatorul receptor este un notebook (rom.: carnet de notiţe) care ar putea fi furat. Stocarea tuturor fişierelor critice în formă criptată reduce mult problemele datorate unor scurgeri de informaţii în eventualitatea în care calculatorul ajunge în mâini nepotrivite.

Cu toate acestea, fişierele de pe disc sunt deseori accesate în ordine aleatoare, în special fişierele din baze de date. Cu o criptare de fişiere care foloseşte înlănţuire de blocuri cifrate, accesul la un bloc aleator necesită întâi decriptarea tuturor blocurilor dinaintea lui, ceea ce reprezintă o soluţie

Cutie de costisitoare. Din acest motiv s-a inventat încă un mod, **modul contor** (eng.: **Counter Mode**), schiţat în fig. 8-15. Aici textul clar nu este criptat direct. În schimb, se criptează vectorul de iniţializare plus o constantă şi textul cifrat rezultat se combină prin XOR cu textul clar. Crescând cu 1 vectorul de iniţializare pentru fiecare bloc nou este uşor să decriptezi un bloc de oriunde din fişier fără a fi nevoie să îi decriptezi toţi predecesorii.

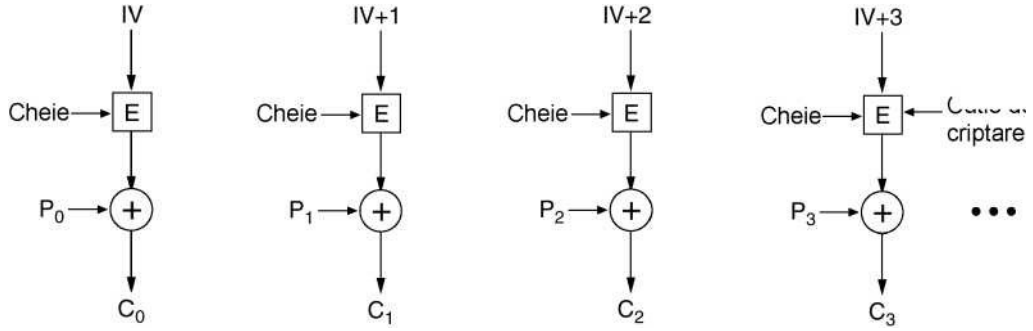


Fig. 8-15. Criptarea folosind modul contor

Deşi modul contor este folositor, el are o slăbiciune care merită să fie evidenţiată. S ă presupunem că aceeaşi cheie, K, este folosită din nou ulterior (cu un text clar diferit dar cu acelaşi IV) şi un atacator obţine tot textul cifrat în ambele rulări. Lanţurile de chei sunt aceleaşi în ambele cazuri, expunând cifrul la un atac la refolosirea lanţului de chei similar celui de la cifrurile înlănţuite. Tot ce are criptanalistul de făcut este XOR între cele două texte cifrate pentru a elimina toată protecţia criptografică şi a obţine combinaţia XOR între textele clare. Această slăbiciune nu înseamnă că modul contor este o idee proastă. Înseamnă doar că şi cheile şi vectorii de iniţializare ar trebui alese independent şi aleator. Chiar dacă aceeaşi cheie este folosită accidental de două ori, dacă vectorul IV este diferit de fiecare dată, textul clar este în siguranţă.

8.2.4 Alte cifruri

DES şi Rijndael sunt cei mai cunoscuţi algoritmi criptografici cu cheie simetrică. Cu toate acestea merită să menţionăm că au fost proiectate numeroase alte cifruri cu cheie simetrică. Câteva dintre acestea sunt integrate cu diverse produse. Câteva dintre cele mai comune sunt enumerate în fig. 8-16.

Cifru	Autor	Lungimea	Comentarii
Blowfish	Bruce Schneier	1 – 448 de biţi	vechi şi lent
DES	IBM	56 de biţi	prea slab pentru a mai fi folosit
IDEA	Massey şi Xuejia	128 de biţi	bun, dar patentat
RC4	Ronald Rivest	1 – 2048 de biţi	atentie: anumite chei sunt
RC5	Ronald Rivest	128 – 256 de	bun, dar patentat
Rijndael	Daemen and Rijmen	128 – 256 de	cea mai bună alegere
Serpent	Anderson, Biham,	128 – 256 de	foarte puternic
Triplu DES	IBM	168 de biţi	a doua alegere
Twofish	Bruce Schneier	128 – 256 de	foarte puternic, larg folosit

Fig. 8-16. Câţiva algoritmi criptografici comuni cu cheie simetrică.

8.2.5 Criptanaliza

Înainte de a părăsi subiectul criptografiei cu chei simetrice, merită cel puţin să menţionăm patru rezultate recente în criptanaliză. Primul este **criptanaliza diferenţială** (Biham şi Shamir, 1993). Această tehnică poate fi folosită pentru a ataca orice cifru bloc. Lucrează la început cu o pereche de blocuri de text clar care diferă doar printr-un număr mic de biţi şi studiază cu grijă ceea ce se întâmplă la fiecare iteraţie internă pe măsură ce criptarea avansează. În multe cazuri, anumite combinaţii apar mai des decât altele şi această observaţie conduce la un atac probabilistic.

Al doilea rezultat demn de notat este **criptanaliza liniară** (Matsui, 1994). Aceasta poate sparge DES-ul cu doar 2^{43} texte clare cunoscute. Lucrează prin combinarea XOR a anumitor biţi din textul clar şi din textul cifrat şi examinarea rezultatelor pentru a descoperi tiparele. Când aceasta se face repetat, jumătate din biţi trebuie să fie 0 şi jumătate să fie 1. Totuşi, adeseori, cifrurile introduc o deviaţie într-o direcţie sau în alta şi această deviaţie, cu toate că este mică, poate fi exploatată pentru a reduce efortul. Pentru mai multe detalii a se vedea lucrarea lui Matsui.

Al treilea rezultat este folosirea analizei consumului de energie electrică pentru a afla chei secrete. Calculatoarele folosesc în mod tipic 3 volţi pentru a reprezenta un bit 1 şi 0 volţi pentru a reprezenta un bit 0. De aceea, procesarea unui 1 consumă mai multă energie electrică decât procesarea unui 0. Dacă un algoritm criptografic constă într-o buclă în care biţii cheii sunt procesaţi în ordine, un atacator care înlocuieşte ceasul principal de *n*-GHz cu un ceas lent (de exemplu 100-Hz) şi pune mufe crocodil pe sursa procesorului şi pinii de masă poate să monitorizeze precis puterea consumată de fiecare instrucţiune maşină. Din aceste date, deducerea cheii este surprinzător de uşoară. Acest tip de criptanaliză poate fi contracarat numai de codificarea atentă a algoritmului în limbaj de asamblare pentru a fi sigur că consumul de putere este independent de cheie şi de asemenea independent de toate cheile de rundă individuale.

Al patrulea rezultat este analiza întârzierilor. Algoritmii criptografici sunt plini de instrucţiuni **if** care testează biţii cheilor de rundă. Dacă părţile **then** şi **else** durează timpi diferiţi, prin încetinirea ceasului şi măsurarea duratei diverşilor paşi se pot deduce cheile de rundă. Odată ce se cunosc toate cheile de rundă, de regulă se poate calcula cheia originală. Analizele de putere şi de întârzieri pot fi folosite simultan pentru a face munca mai uşoară. Cu toate că analizele de putere şi de întârzieri pot părea exotice, în realitate ele sunt tehnici puternice care pot sparge orice cifru care nu a fost proiectat în mod special pentru a le rezista.

8.3 ALGORITMI CU CHEIE PUBLICĂ

Istoric, distribuţia cheilor a fost întotdeauna punctul slab al multor criptosisteme. Indiferent de cât de puternic era un criptosistem, dacă un intrus

putea fura cheia, sistemul îşi pierdea valoarea. Criptologii au considerat întotdeauna ca de la sine înţeles faptul că atât pentru criptare cât şi pentru decriptare se foloseşte aceeaşi cheie (sau una uşor derivabilă din cealaltă). Dar cheia trebuia distribuită tuturor utilizatorilor sistemului. Astfel, părea a exista întotdeauna următoarea problemă inerentă: cheile trebuia protejate contra furtului dar, în acelaşi timp, ele trebuiau distribuite, astfel încât ele nu puteau fi sechestrate într-un seif de bancă.

În 1976, doi cercetători de la Universitatea Stanford, Diffie şi Hellman (1976), au propus un tip radical nou de criptosistem în care cheile de criptare şi decriptare sunt diferite, iar cheia de decriptare nu poate fi dedusă din cheia de criptare. În propunerea lor, algoritmul (cheia) de criptare, E, şi algoritmul (cheia) de decriptare, D, trebuiau să satisfacă trei cerinţe. Aceste cerinţe pot fi exprimate simplificat după cum urmează:

1. $D(E(P))=P$
2. Este mai mult decât dificil să se deducă D din E.
3. E nu poate fi spart printr-un atac cu text clar ales.

Prima cerinţă spune că, dacă se aplică D unui mesaj criptat, $E(P)$, se obţine textul clar original, P. Fără această proprietate, receptorul legitim nu ar putea decripta textul cifrat. Cea de-a doua cerinţă vorbeşte de la sine. Cea de-a treia cerinţă este necesară deoarece, după cum vom vedea curând, intruşii pot experimenta şi testa algoritmul după pofta inimii. În aceste condiţii, nu există nici un motiv pentru ca E, cheia de criptare, să nu poată fi făcută publică.

Metoda lucrează astfel: o persoană, să spunem Alice, dorind să primească mesaje secrete, concepe mai întâi doi algoritmi ce satisfac cerinţele de mai sus. Algoritmul de criptare şi cheia lui Alice sunt făcuţi apoi publici, de unde şi numele de **criptografie cu cheie publică**. Alice poate de exemplu să îşi pună cheia publică pe pagina ei personală de pe Web. Vom folosi notaţia E_A pentru algoritmul de criptare parametrizat de cheia publică a lui Alice. În mod similar, algoritmul (secret) de decriptare parametrizat de cheia privată a lui Alice este D_A. Bob face acelaşi lucru, publicând E_B, dar ţinând D_B secret.

Să vedem acum dacă putem rezolva problema stabilirii unui canal sigur între Alice şi Bob, care nu au mai avut niciodată vreun contact anterior. Atât cheia de criptare a Alicei, E_A, cât şi cea a lui Bob, E_B, sunt presupuse a se găsi într-un fişier ce poate fi citit de oricine. Acum Alice ia primul ei mesaj, P, calculează $E_B(P)$ şi îl trimite lui Bob. Bob îl decriptează aplicându-i cheia sa secretă D_B [adică, el calculează $D_B(E_B(P))=P$]. Nimeni altcineva nu poate citi mesajul criptat, $E_B(P)$, deoarece sistemul de criptare este presupus puternic şi deoarece este prea greu să se deducă $D_B(P)$ din $E_B(P)$ public cunoscut. Pentru a trimite un răspuns R, Bob transmite $E_A(R)$. Alice şi Bob pot comunica acum într-o manieră sigură.

În acest punct ar fi poate utilă o observaţie asupra terminologiei. Criptografia cu cheie publică necesită ca fiecare utilizator să aibă două chei: o cheie publică, folosită de toată lumea pentru a cripta mesajele ce-i sunt trimise, şi o cheie secretă, de care utilizatorul are nevoie ca să-şi decripteze mesajele. Ne vom referi în mod

constant la aceste chei ca fiind cheia *publică* şi, respectiv, cheia *privată* şi le vom deosebi de cheile *secrete* folosite pentru criptografia convenţională cu cheie simetrică.

8.3.1 RSA

Singura problemă este aceea că avem nevoie de algoritmi care să satisfacă complet toate cele trei cerinţe. Datorită posibilelor avantaje ale criptografiei cu chei publice, mulţi cercetători au lucrat din greu la acest subiect şi au fost deja publicaţi câţiva algoritmi. O metodă bună a fost descoperită de un grup de la MIT (Rivest et. al, 1978). Ea este cunoscută prin iniţialele numelor celor trei descoperitori (Rivest, Shamir, Adelman): **RSA**. Metoda a supravieţuit tuturor încercărilor de a o sparge timp de mai mult de un sfert de secol şi este considerată foarte puternică. Multe aplicaţii de securitate se bazează pe ea. Dezavantajul major al acesteia este că necesită chei de cel puţin 1024 de biţi pentru o securitate bună (spre deosebire de 128 biţi pentru algoritmii cu cheie simetrică), ceea ce o face destul de lentă.

Metoda RSA este bazată pe câteva principii din teoria numerelor. Vom rezuma mai jos modul în care se foloseşte această metodă; pentru detalii, a se consulta articolul.

1. Se aleg două numere prime, p şi q, (de obicei de 1024 biţi).
2. Se calculează $n = p \times q$ şi $z = (p - 1) \times (q - 1)$.
3. Se alege un număr relativ prim cu z şi este notat cu d.
4. Se găseşte e astfel încât $e \times d = 1 \bmod z$.

Cu aceşti parametri calculaţi în avans, suntem gata să începem criptarea. Împărţim textul clar (privit ca şir de biţi) în blocuri, astfel încât fiecare mesaj de text clar, P, să intre în intervalul $0 < P < n$. Aceasta poate fi făcută grupând textul clar în blocuri de câte k biţi, unde k este cel mai mare număr întreg pentru care inegalitatea $2^k < n$ este adevărată.

Pentru a cripta mesajul P, se calculează $C = P^e \pmod n$. Pentru a decripta C, se calculează $P = C^d \pmod n$. Se poate demonstra că pentru toţi P din intervalul specificat, criptarea şi decriptarea sunt funcţii inverse una alteia. Pentru a realiza criptarea este nevoie de e şi n. Pentru a realiza decriptarea este nevoie de d şi n. De aceea, cheia publică constă din perechea (e, n) iar cheia privată din perechea (d, n).

Securitatea metodei este bazată pe dificultatea factorizării numerelor mari. Dacă un criptanalist ar putea factoriza numărul n (public cunoscut), el ar putea găsi p şi q, iar din acestea pe z. Cu z şi e cunoscuţi, criptanalistul îl poate calcula pe d folosind algoritmul lui Euclid. Din fericire, matematicienii au încercat să factorizeze numere mari de cel puţin 300 de ani şi experienţa acumulată sugerează că aceasta este o problemă mai mult decât dificilă.

După Rivest şi colegii săi, factorizarea unui număr de 500 de cifre necesită un timp de calcul de 1025 ani folosind forţa brută. În ambele cazuri ei presupun că se foloseşte cel mai bun algoritm de factorizare şi un calculator cu timp de execuţie a unei instrucţiuni de 1 psec. Chiar dacă viteza calculatoarelor va continua să sporească cu un ordin de mărime pe deceniu, vor mai trece secole până când factorizarea unui număr de 500 de cifre va deveni realizabilă, moment în care

descendenţii noştri vor alege pur şi simplu p şi q mai mari.

Un exemplu didactic banal pentru algoritmul RSA este dat în fig. 8-17. Pentru acest exemplu am ales $p = 3$ şi $q = 11$, rezultând $n = 33$ şi $z = 20$. O valoare potrivită pentru d este $d = 7$, deoarece 7 şi 20 nu au factori comuni. Cu aceste alegeri, e poate fi găsit prin rezolvarea ecuaţiei $7e = 1 \pmod{20}$, care dă $e = 3$. Textul cifrat, C, pentru textul clar al mesajului, P, este dat de $C = P^3 \pmod{33}$. Textul cifrat este decriptat de către receptor după regula $P = C^7 \pmod{33}$. Fig. prezintă ca exemplu criptarea şi decriptarea textului clar „SUZANNE”.

Text clar (P)			Text cifrat (C)		După decriptare	
Simbolic	Numeric	P^3	$P^3 \pmod{33}$	C^7	$C^7 \pmod{33}$	Simbolic
S	19	6859	28	13492928512	19	S
U	21	9261	21	1801088541	21	U
z	26	17576	20	1280000000	26	Z
A	01	1	1	1	01	A
N	14	2744	5	78125	14	N
N	14	2744	5	78125	14	N
E	05	125	26	8031810176	05	E
Calcul efectuat de emiţător			Calcul efectuat de receptor			

Fig. 8-17. Un exemplu de algoritm RSA.

Deoarece numerele prime alese pentru acest exemplu sunt prea mici, P trebuie să fie mai mic decât 33, deci fiecare bloc de text clar poate conţine doar un singur caracter. Rezultatul este un cifru cu substituţie monoalfabetică, nu foarte impresionant. Dacă în locul acestora am fi ales p şi $q = 2^{512}$, am fi avut $n = 2^{1024}$, astfel încât fiecare bloc poate fi de până la 1024 de biţi sau 128 de caractere de 8 biţi, faţă de 8 caractere pentru DES şi 16 caractere pentru AES.

Trebuie subliniat că folosirea RSA în modul descris este similară folosirii unui algoritm simetric în modul ECB – blocuri de intrare identice conduc la blocuri de ieşire identice. De aceea este necesară o anumită formă de înlănţuire pentru criptarea datelor. Totuşi, în practică, cele mai multe sisteme bazate pe RSA folosesc criptografia cu cheie publică în principal pentru distribuirea cheilor de sesiune de unică folosinţă utilizate pentru un algoritm simetric ca AES sau triplu DES. RSA este prea lent pentru a cripta eficient volume mari de date, dar este folosit mult la distribuţia de chei.

8.3.2 Alţi algoritmi cu cheie publică

Cu toate că RSA este larg răspândit, nu este în nici un caz singurul algoritm cu cheie publică cunoscut. Primul algoritm cu cheie publică a fost algoritmul rucsacului (Merkle şi Hellman, 1978). Ide- ea este că cineva posedă un număr mare de obiecte, fiecare cu greutate diferită. Posesorul codifică mesajul prin selecţia secretă a unei submulţimi de obiecte şi plasarea lor în rucsac. Greutatea totală a obiectelor din rucsac este făcută publică, ca şi lista tuturor obiectelor posibile. Lista obiectelor din rucsac este ţinută secretă. Cu câteva restricţii suplimentare, problema găsirii unei liste de obiecte cu greutatea dată a fost gândită ca fiind intractabilă computaţional şi formează baza pentru algoritmul cu cheie publică.

Inventatorul algoritmului, Ralph Merkle, a fost aproape sigur că acest algoritm

nu poate fi spart, astfel că el a oferit o recompensă de 100 de dolari oricui îl va putea sparge. Adi Shamir („S”-ul din RSA) l-a spart cu promptitudine şi a primit recompensa. Fără a-şi pierde curajul, Merkle şi-a întărit algoritmul şi a oferit o recompensă de 1000 de dolari oricui va putea sparge noul algoritm. Ronald Rivest („R” -ul din RSA) l-a spart cu promptitudine şi a luat banii. Merkle nu a îndrăznit să ofere 10000 de dolari pentru următoarea versiune, astfel că „A” (Leonard Adelman) nu a avut noroc. Cu toate acestea, algoritmul rucsacului nu este considerat sigur şi este rareori utilizat.

Alte scheme cu cheie publică sunt bazate pe dificultatea calculului logaritmilor discreţi (Rabin, 1979). Algoritmii care folosesc acest principiu au fost inventaţi de El Gamal (1985) şi Schnorr (1991).

Există câteva alte scheme, cum ar fi cele bazate pe curbe eliptice (Menezes şi Vanstone, 1993), dar cele două categorii majore sunt cele bazate pe dificultatea factorizării numerelor mari şi a calculului logaritmilor discreţi modulo un număr prim mare. Aceste probleme sunt considerate ca fiind cu adevărat dificile deoarece matematicienii le studiază de mulţi ani fără vreun progres notabil.

8.4 SEMNĂTURI DIGITALE

Autenticitatea multor documente legale, financiare şi de alt gen este determinată de prezenţa sau absenţa unor semnături autorizate scrise de mână. Iar fotocopiile nu sunt valabile. Pentru ca sistemele de mesaje computerizate să înlocuiască transportul fizic al documentelor scrise cu cerneală pe hârtie trebuie găsită o metodă ca documentele să fie semnate într-un mod nefalsificabil.

Problema de a concepe un înlocuitor pentru semnăturile scrise de mână este destul de dificilă. De fapt, este necesar un sistem prin care una din părţi poate trimite mesaje „semnate” celeilalte părţi astfel încât:

1. Receptorul poate verifica identitatea pe care pretinde a o avea transmiţătorul;
2. Transmiţătorul nu poate să nege mai târziu că e autorul mesajului;
3. Receptorul nu poate să fi pregătit el însuşi mesajul.

Prima cerinţă este necesară, de exemplu, în sistemele financiare. Atunci când calculatorul unui client ordonă calculatorului unei bănci să cumpere o tonă de aur, calculatorul băncii trebuie să poată să se asigure că acel calculator care dă ordinul aparţine într-adevăr companiei al cărei cont va fi debitat. Cu alte cuvinte, banca trebuie să autentifice clientul (şi clientul trebuie să autentifice banca).

A doua cerinţă este necesară pentru a proteja banca împotriva fraudei. Să presupunem că banca cumpără o tonă de aur şi imediat după aceea preţul aurului scade brusc. Un client necinstit poate să acuze banca, pretinzând că el nu a emis niciodată vreun ordin de cumpărare de aur. Când banca prezintă mesajul în faţa curţii, clientul neagă faptul că l-ar fi trimis. Proprietatea că nici o parte a unui contract nu poate nega mai târziu faptul că l-a semnat se numeşte **nerepudiere** (eng.:

nonrepudiation). Schemele cu semnătură digitală pe care le vom studia acum oferă această proprietate.

Cea de-a treia cerinţă este necesară pentru a proteja clientul în eventualitatea că preţul aurului explodează şi banca încearcă să construiască un mesaj în care clientul cere cumpărarea unui lingou de aur în locul unei tone. În acest scenariu de fraudă, banca îşi păstrează restul de aur pentru ea însăşi.

8.4.1 Semnături cu cheie simetrică

Un mod de abordare pentru semnăturile digitale este acela de a avea o autoritate centrală care ştie totul şi în care oricine are încredere, să spunem Big Brother *(BB,* rom.: fratele cel mare). Fiecare utilizator alege o cheie secretă şi o duce personal la biroul *BB*. Astfel, doar Alice şi *BB* vor cunoaşte cheia secretă a lui Alice, K_A, ş.a.m.d.

Atunci când Alice doreşte să trimită un mesaj în clar semnat, *P,* bancherului său, Bob, ea generează $K_A(B, R_A, t, P)$, unde *B* este identitatea lui Bob, R_A este un număr aleator ales de Alice, *t* este o amprentă de timp pentru a asigura prospeţimea şi $K_A(B, R_A, t, P)$ este mesajul criptat cu cheia ei K_A. Apoi îl trimite, după cum este se arată şi în fig. 8-18. *BB* vede că mesajul este de la Alice, îl decriptează şi îi trimite lui Bob mesajul ca în figură. Mesajul trimis spre Bob conţine textul clar din mesajul lui Alice şi, de asemenea, mesajul semnat $K_{BB}(A, t, P)$, unde t este amprenta de timp. Acum Bob rezolvă cererea lui Alice.

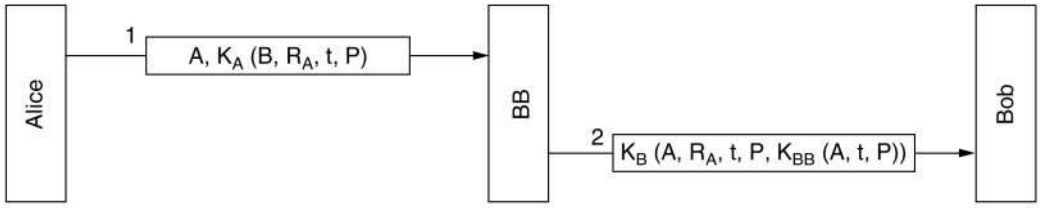


Fig. 8-18. Semnături digitale cu Big Brother.

Ce se întâmplă dacă Alice neagă mai târziu trimiterea mesajului? Primul pas este că toată lumea dă în judecată pe toată lumea (cel puţin în S.U.A.). În final, când cazul ajunge în faţa curţii şi Alice neagă cu înverşunare că a trimis lui Bob mesajul în dispută, judecătorul îl va întreba pe Bob cum poate fi sigur că mesajul disputat vine de la Alice şi nu de la Trudy. Bob va arăta mai întâi că *BB* nu acceptă un mesaj de la Alice decât dacă este criptat cu K_A, aşa că nu există nici o posibilitate ca Trudy să-i trimită lui *BB* un mesaj fals ca provenind de la Alice fără ca BB să îl detecteze imediat.

Apoi Bob va aduce în mod incontestabila Probă A, $K_{BB}(A, t, P)$. Bob spune că acesta este un mesaj semnat de *BB* care demonstrează că Alice i-a trimis *P* lui Bob. Judecătorul îi va cere apoi lui *BB* (în care toată lumea are încredere) să decripteze Proba A. Când *BB* va depune mărturie că Bob spune adevărul, judecătorul va de verdictul în favoarea lui Bob. Cazul va fi închis.

O problemă posibilă cu protocolul de semnare din fig. 8-18 apare atunci când Trudy replică oricare din mesaje. Pentru a minimiza această problemă, sunt folosite peste tot amprente de timp. Mai mult, Bob poate verifica toate mesajele recente să vadă dacă R_A a fost folosit în vreunul dintre ele. Dacă da, mesajul respectiv este ignorat deoarece este o replică. De remarcat că Bob va refuza toate mesajele foarte vechi din punct de vedere al amprentei de timp. Pentru a se păzi împotriva atacurilor cu replică instantanee, Bob verifică doar R_A al oricărui mesaj venit, ca să vadă dacă s-a mai primit în ultima oră un astfel de mesaj de la Alice. Dacă nu, Bob poate presupune fără nici un risc că mesajul reprezintă o nouă cerere.

8.4.2 Semnături cu cheie publică

O problemă structurală în folosirea criptografiei cu cheie secretă pentru semnături digitale este aceea că oricine trebuie să se încreadă în Big Brother. Mai mult decât atât, Big Brother poate citi toate mesajele semnate. Cei mai logici candidaţi pentru a juca rolul lui Big Brother sunt guvernul, băncile, contabilii şi avocaţii. Din păcate, nici una dintre aceste organizaţii nu inspiră încredere totală tuturor cetăţenilor. De aceea ar fi frumos dacă semnarea documentelor nu ar necesita existenţa unei astfel de autorităţi de încredere.

Din fericire, criptografia cu cheie publică îşi poate aduce aici o importantă contribuţie. Să presupunem că algoritmii de criptare şi decriptare cu cheie publică au proprietatea că $E(D(P))= P$ în plus faţă de proprietatea uzuală $D(E(P))=P$. (RSA are această proprietate, deci presupunerea nu este nerezonabilă). Presupunând acest lucru, Alice poate trimite un text clar semnat, P, lui Bob transmiţând $E_B(D_A(P))$. O observaţie importantă aici este aceea că Alice cunoaşte atât propria sa cheie secretă, D_A, cât şi cheia publică a lui Bob, E_B, astfel încât construcţia acestui mesaj este pentru Alice un lucru realizabil.

Când Bob primeşte mesajul, el îl transformă, folosindu-şi cheia privată, ca de obicei, rezultând $D_A(P)$, după cum este arătat şi în fig. 8-19. El memorează acest text într-un loc sigur şi apoi aplică E_A pentru a obţine textul clar original.

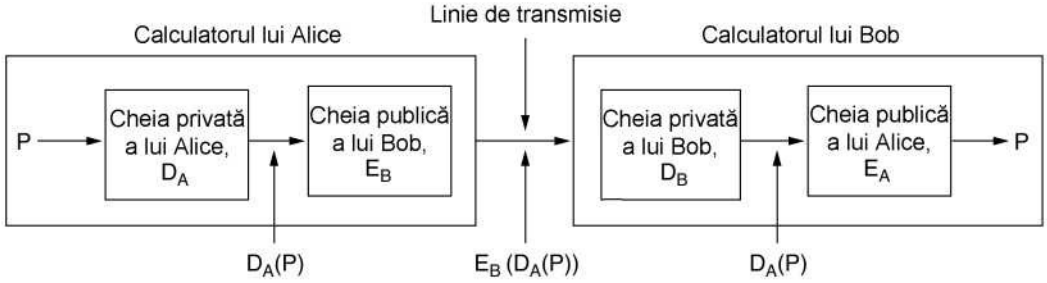


Fig. 8-19. Semnături digitale folosind criptografia cu cheie publică.

Pentru a vedea cum lucrează proprietatea de semnătură, să presupunem că Alice neagă ulterior trimiterea mesajului P lui Bob. Atunci când cazul ajunge în faţa curţii, Bob poate aduce ca probe atât P cât şi $D_A(P)$. Judecătorul poate verifica uşor că Bob are un mesaj valid criptat cu D_A, doar aplicând E_A asupra lui. Deoarece Bob nu ştie care este cheia privată a lui Alice, singurul mod prin care Bob poate să fi primit mesajul criptat cu această cheie privată este ca Alice în persoană să-l fi

trimis. Cât timp va sta în închisoare pentru minciună şi fraudă Alice va avea suficient timp să conceapă noi algoritmi cu cheie publică interesanţi.

Cu toate că folosirea criptografiei cu cheie publică pentru semnături digitale este o schemă elegantă, există probleme legate mai degrabă de mediul în care acestea operează decât de algoritmul de la bază. De exemplu, Bob poate dovedi că mesajul a fost trimis de către Alice doar atâta vreme cât DA rămâne secret. Dacă Alice dezvăluie cheia sa secretă acest argument nu va mai fi valabil, deoarece oricine poate să fi trimis mesajul, chiar şi Bob.

Problema poate apărea, de exemplu, dacă Bob este agentul de vânzări al lui Alice. Alice îi spune lui Bob să cumpere nişte acţiuni. Imediat după aceea, preţul scade vertiginos. Pentru a repudia mesajul său către Bob, Alice face o plângere la poliţie, pretinzând că i-a fost spartă casa şi furată cheia secretă. În funcţie de legislaţia din ţara sau ţinutul său, ea poate fi sau nu răspunzătoare legal, în special dacă pretinde că a descoperit spargerea când s-a întors acasă de la muncă, la câteva ore mai târziu.

O altă problemă cu schema de semnătură este ce se întâmplă dacă Alice decide să-şi schimbe cheia. A face acest lucru este evident legal şi este probabil o idee bună să o facă periodic. Dacă în justiţie apare mai târziu un caz, aşa cum s-a povestit mai sus, judecătorul va aplica actualul EA la DA(P) şi va descoperi că nu se obţine P. Bob va fi atunci într-o situaţie delicată.

În principiu, orice algoritm cu cheie publică poate fi folosit pentru semnături digitale. Standardul de facto în industrie este algoritmul RSA. Multe produse pentru securitate îl folosesc. Totuşi, în 1991, NIST (National Institute of Standards and Technology) a propus o variantă a algoritmului cu cheie publică El Gamal pentru noul lor standard **DSS** (**Digital Signature Standard,** rom.: Standard pentru Semnătură Digitală) . El Gamal îşi bazează securitatea pe dificultatea calculului logaritmilor discreţi şi nu pe dificultatea factorizării numerelor mari.

Ca de obicei, când guvernul încearcă să impună standarde criptografice, s-a iscat o reacţie antagonistă de masă. DSS a fost criticat pentru a fi:

1. Prea secret (NSA a proiectat protocolul pentru folosirea El Gamal).
2. Prea lent (de 10 până la 40 de ori mai lent decât RSA în verificarea semnăturilor).
3. Prea nou (El Gamal nu a fost încă suficient analizat).
4. Prea nesigur (chei fixe de 512 biţi).

După o revizuire ulterioară, cel de-al patrulea motiv a fost eliminat fiindcă s-au permis chei de până la 1024 biţi. Cu toate acestea, primele două puncte rămân valide.

8.4.3 Rezumate de mesaje

O critică adusă schemelor de semnătură este aceea că adeseori cuplează două funcţii distincte: autentificare şi confidenţialitate. Adesea, autentificarea este necesară, dar confidenţialitatea nu. De asemenea, obţinerea unei licenţe de export este deseori mai uşoară dacă sistemul în chestiune oferă numai autentificare dar nu şi confidenţialitate. Mai jos vom descrie o schemă de autentificare care nu necesită

criptarea întregului mesaj.

Schema este bazată pe ideea unei funcţii de dispersie neinversabile care preia o bucată de text clar de lungime arbitrară din care calculează un şir de biţi de lungime fixă. Funcţia de dispersie, *MD,* adeseori numită **rezumat** (**digest**) **al mesajului**, are patru proprietăţi importante:

1. Dat fiind *P,* este uşor de calculat *MD(P).*
2. Dat fiind *MD*(*P*), este efectiv imposibil de calculat *P*.
3. Dat fiind *P* nimeni nu poate găsi *P* astfel încât *MD(P)=MD(P)*.
4. O schimbare la intrare chiar şi de 1 bit produce o ieşire foarte diferită.

Pentru a satisface criteriul 3, dispersia trebuie să aibă cel puţin 128 de biţi lungime, de preferat chiar mai mult. Pentru a satisface criteriul 4, dispersia trebuie să amestece biţii foarte bine, la fel ca algoritmii de criptare cu cheie simetrică pe care i-am văzut.

Calculul rezumatului unui mesaj dintr-o bucată de text clar este mult mai rapidă decât criptarea acelui text clar cu un algoritm cu cheie publică, deci rezumatele de mesaje pot fi folosite pentru a oferi viteză algoritmilor cu semnătură digitală . Pentru a vedea cum lucrează, să considerăm din nou protocolul de semnătură din fig. 8-18. În loc de a semna P cu $K_{BB}(A, t, P)$, *BB* calculează acum rezumatul mesajului aplicând *MD* lui *P*, rezultând *MD*(*P*). *BB* include apoi $K_{BB}(A, t, MD(P))$ ca al cincilea element în lista criptată cu K_B care este trimisă lui Bob, în loc de $K_{BB}(A, t, P)$.

Dacă apare o dispută, Bob poate aduce ca argumente atât *P* cât şi $K_{BB}(A, t, MD(P))$. După ce Big Brother l-a decriptat pentru judecător, Bob are *MD*(*P*), care este garantat a fi original şi pretinsul *P*. Totuşi, deoarece este efectiv imposibil ca Bob să găsească un alt mesaj care să aibă acest rezumat, judecătorul va fi convins uşor că Bob spune adevărul. Folosirea rezumatelor mesajelor în acest mod economiseşte atât timpul de criptare cât şi costurile pentru transport şi memorare.

Calculul rezumatelor mesajelor funcţionează şi în criptosistemele cu chei publice, după cum este arătat şi în fig. 8-20. Aici, Alice calculează mai întâi rezumatul de mesaj pentru textul său clar. Apoi ea semnează rezumatul şi trimite atât rezumatul semnat cât şi textul clar lui Bob. Dacă Trudy îl înlocuieşte pe *P* în timpul transferului, Bob va vedea aceasta atunci când va calcula el însuşi *MD*(*P*).

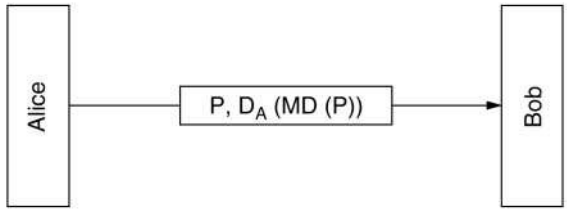


Fig. 8-20. Semnături digitale folosind rezumatul mesajului.

MD5

Au fost propuse diverse de funcţii pentru calculul rezumatului mesajelor. Cele mai folosite sunt MD5 (Rivest, 1992) şi SHA-l (NIST, 1993). **MD5** este a cincea dintr-o serie de funcţii de dispersie proiectate de Ronald Rivest. Operează prin amestecarea biţilor într-un mod suficient de complicat, astfel încât fiecare bit de

ieşire să fie afectat de fiecare bit de intrare. Foarte pe scurt, algoritmul începe prin a umple mesajul până la o lungime de 448 de biţi (modulo 512). Apoi lungimea originală a mesajului este adăugată ca un întreg pe 64 de biţi, dând o intrare totală a cărei lungime este un multiplu de 512 de biţi. Ultimul pas dinaintea începerii calculului este iniţializarea unui tampon de 128 de biţi la o valoare fixă .

Apoi începe calculul. Fiecare rundă ia un bloc de intrare de 512 de biţi şi îl amestecă complet cu tamponul de 128 de biţi. Ca o măsură suplimentară, este folosită şi o tabelă construită folosind funcţia sinus. Utilizarea unei funcţii cunoscute cum este sinus nu se datorează faptului că este mai aleatoare decât un generator de numere aleatoare, ci pentru a evita orice suspiciune că proiectantul a inclus o trapă inteligent ascunsă prin care doar el poate intra. Refuzul IBM-ului de a dezvălui principiile aflate la baza proiectării cutiilor S din DES a dus la destul de multe speculaţii privind existenţa trapelor ascunse. Rivest a vrut să evite această suspiciune. Pentru fiecare bloc de intrare sunt efectuate patru runde. Acest proces continuă până când sunt consumate toate blocurile de intrare. Conţinutul tamponului de 128 de biţi formează rezumatul mesajului.

MD5 este activ deja de peste o decadă şi a fost atacat de mulţi. S-au găsit câteva vulnerabilităţi, dar anumiţi paşi interni previn spargerea lui. Totuşi, dacă barierele care rămân în MD5 cad, poate că într-un final va cădea şi el. Cu toate acestea, la momentul scrierii acestei cărţi el stătea încă în picioare.

SHA-1

Altă funcţie majoră pentru calculul rezumatului este **SHA** (**Secure Hash Algorithm,** rom.: Algoritm de Dispersie Sigur), dezvoltată de NSA şi acceptată de către NIST ca FIPS 180-1. Ca şi MD5, SHA-l prelucrează datele de intrare în blocuri de câte 512 de biţi, dar, spre deosebire de MD5, generează un rezumat de mesaj de 160 de biţi. Un mod tipic pentru Alice de a trimite un mesaj nesecret dar semnat lui Bob este ilustrat în fig. 8-21. Aici mesajul ei de text clar este introdus în algoritmul SHA-l pentru a obţine o dispersie SHA-l de 160 biţi. Apoi Alice semnează dispersia cu cheia sa privată RSA şi trimite atât mesajul în text clar cât şi dispersia semnată lui Bob.

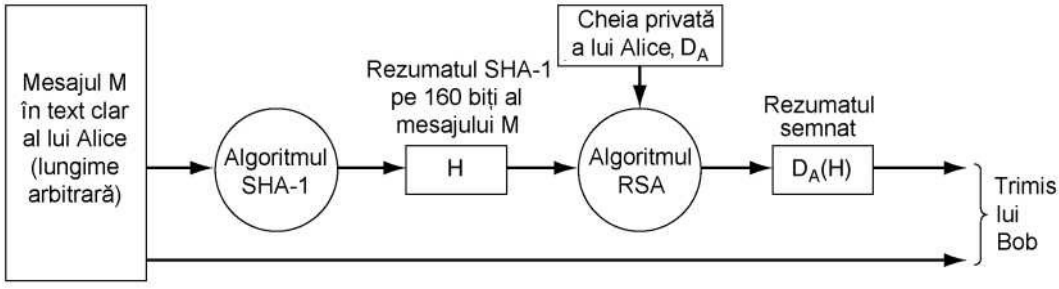


Fig. 8-21. Folosirea lui SHA- 1 şi RSA pentru a semna mesaje nesecrete

După ce primeşte mesajul, Bob calculează el însuşi dispersia SHA- 1 şi aplică de asemenea cheia publică a lui Alice asupra dispersiei semnate pentru a obţine dispersia originală *H.* Dacă cele două se potrivesc, mesajul este considerat valid. Deoarece nu există nici o cale pentru Trudy de a modifica mesajul (în text clar) în timp ce se trimite şi de a produce unul nou care dă ca dispersie H, Bob poate să

detecteze uşor orice modificări pe care le-a adus Trudy mesajului. Pentru mesajele a căror integritate este importantă dar al căror conţinut nu este secret, se foloseşte pe larg schema din fig. 8-21. În schimbul unui cost de calcul relativ mic, ea garantează că orice modificări făcute asupra mesajului în text clar în tranzit pot fi detectate cu o probabilitate foarte mare.

Acum să vedem pe scurt cum lucrează SHA-1. Acesta începe prin a completa mesajul, adăugând la sfârşit un bit 1, urmat de atâţi biţi 0 câţi sunt necesari pentru a obţine o lungime multiplu de 512 de biţi. Apoi se introduce prin OR în cei 64 de biţi mai puţin semnificativi un număr de 64 biţi conţinând lungimea mesajului înaintea completării. În fig. 8-22, este arătat mesajul cu completare la dreapta deoarece textul şi cifrele englezeşti merg de la stânga la dreapta (adică dreapta jos este perceput în general ca sfârşit de cifră). La calculatoare, această orientare corespunde la maşini de tip

big-endian ca SPARC, dar SHA-l completează întotdeauna sfârşitul mesajului, indiferent de ce tip de endian este maşina.

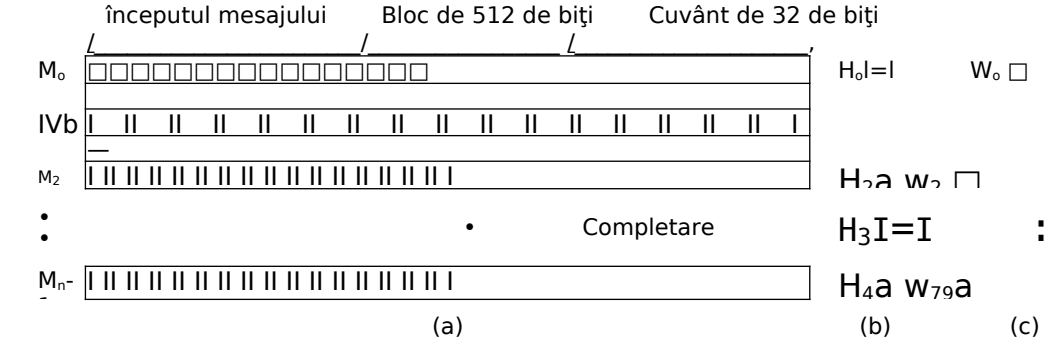


Fig. 8-22. (a) Un mesaj completat până la un multiplu de 512 biţi. (b) Variabilele de ieşire. (c)
Vectorul de cuvinte

În timpul calculului, SHA-l păstrează cinci variabile de 32 de biţi, H_o până la H_4, în care se acumulează dispersia. Acestea sunt arătate în fig. 8-22(b). Ele sunt iniţializate la constante specificate în standard.

Fiecare din blocurile M_o până la M_{n-l} este prelucrat pe rând. Pentru blocul curent, primele 16 cuvinte sunt copiate mai întâi la începutul unui vector auxiliar de 80 de cuvinte, W, ca în fig. 8-22(c). Apoi celelalte 64 de cuvinte din W sunt umplute folosind formula

$$W_i = S^i(W_{i-3} \text{ XOR } W_{i-8} \text{ XOR } W_{i-14} \text{ XOR } W^{\wedge}) \quad (16 < i < 79)$$

unde $S^b(W)$ reprezintă rotaţia circulară la stânga, cu *b* biţi a cuvântului de 32 de biţi *W.* Cinci variabile auxiliare, de la A până la E, sunt apoi iniţializate din H_o – H_4 respectiv.

Calculul poate fi prezentat în pseudo-C astfel:

```
for (i = 0; i < 80; i++) {
        temp = S^5(A) + f_l(B, C, D) + E + W_i + K_i;
        E = D; D = C; C = S^30(B); B = A; A = temp;
}
```

unde constantele K_i sunt definite în standard. Funcţiile de amestecare f_i sunt definite astfel:

$f_i(B, C, D) = (B \text{ AND } C) \text{ OR } (\text{NOT } B \text{ AND } D)$ $(0 < i > 19)$

$f_i(B, C, D) = B \text{ XOR } C \text{ XOR } D$ $(20 < i > 39)$

$f_i(B, C, D) = (B \text{ AND } C) \text{ OR } (B \text{ AND } D) \text{ OR } (C \text{ AND } D)$ $(40 < i > 59)$

$f_i(B, C, D) = B \text{ XOR } C \text{ XOR } D$ $(60 < i > 79)$

Când toate cele 8o de iteraţii ale buclei sunt terminate, variabilele A până la E sunt adăugate la H_o până la H_4, respectiv.

După prelucrarea primului bloc de 512 de biţi, se începe următorul. Vectorul W este reiniţializat din noul bloc, dar *H* este lăsat cum era. Când acest bloc se termină, se începe următorul, ş.a.m.d., până când toate blocurile de 512 de biţi ale mesajului au fost băgate în seamă. Când ultimul bloc a fost terminat, cele cinci cuvinte de 32 de biţi din vectorul *H* sunt scoase la ieşire ca dispersia criptografică de 160 biţi. Codul complet C pentru SHA-l este dat în RFC 3174.

Noi versiuni ale SHA-l sunt în curs de alcătuire pentru dispersii de respectiv 256, 384 şi 512 biţi.

8.4.4 Atacul zilei de naştere

În lumea criptografiei, nimic nu este ceea ce pare a fi. S-ar putea crede că sunt necesare 2m operaţii pentru a falsifica un rezumat de m biţi. De fapt, 2m/2 sunt suficiente dacă se foloseşte atacul zilei de naştere, o abordare publicată de Yuval (1979) în lucrarea sa „How to Swindle Rabin” (rom.: Cum să-l jefuieşti pe Rabin).

Ideea acestui atac vine de la o tehnică pe care profesorii de matematică o folosesc adeseori în cursurile lor de teoria probabilităţilor. Întrebarea este: Câţi studenţi trebuie să fie într-o clasă pentru ca probabilitatea de a exista doi studenţi cu aceeaşi dată de naştere să fie 1/2? Mulţi studenţi se aşteaptă ca răspunsul să fie ceva peste 100. De fapt, teoria probabilităţilor spune că trebuie să fie exact 23. Fără a face o analiză riguroasă, intuitiv, cu 23 de oameni se pot forma (23x23)/2 = 253 perechi diferite, fiecare dintre ele având o probabilitate de 1/365 să fie cea potrivită. În această lumină, rezultatul nu mai este deloc surprinzător.

Mai general, dacă există o corespondenţă între intrări şi ieşiri cu n intrări (oameni, mesaje etc.) şi k ieşiri (zile de naştere, rezumate etc.), există n(n-2)/2 perechi de intrare. Dacă n(n-2)/2 > k, şansa de a avea cel puţin o potrivire este destul de mare. Astfel, cu aproximaţie, o potrivire este posibilă pentru $^{n} > y[k$. Acest rezultat înseamnă că un rezumat de 64 de biţi poate fi probabil spart prin generarea a aproape 232 mesaje şi căutând două cu acelaşi rezumat.

Să vedem acum un exemplu practic. Departamentul de Ştiinţa Calculatoarelor de la Universitatea de Stat are disponibil un post de profesor titular şi există doi candidaţi la el, Tom şi Dick. Tom a fost angajat cu doi ani înaintea lui Dick, aşa că el merge primul la verificare. Dacă reuşeşte, Dick are ghinion. Tom ştie că şefa departamentului, Marylin, are o impresie bună despre activitatea sa, aşa că o roagă

să îi scrie o scrisoare de recomandare pentru decan, cel care va decide în cazul său. Odată trimise, toate scrisorile devin confidenţiale.

Marylin spune secretarei, Ellen, să scrie Decanului o scrisoare, subliniind ceea ce doreşte. Când este gata, Marylin o va revizui, va calcula şi va semna un rezumat de 64 de biţi şi îl va trimite decanului. Ellen poate trimite scrisoarea mai târziu, prin poştă electronică.

Din nefericire pentru Tom, Ellen are o idilă cu Dick şi i-ar plăcea să îl facă pe Tom să eşueze, aşa că ea scrie o scrisoare cu următoarele 32 de opţiuni cuprinse între paranteze drepte.

Dear Dean Smith,

This *[letter | message]* is to give my *[honest | frank]* opinion of Prof. Tom Wilson, who is *[a candidate | up]* for tenure *[now | this year].* I have *[known | worked with]* Prof. Wilson for *[about | almost]* six years. He is an *[outstanding | excellent]* researcher of great *[talent | ability]* known *[worldwide | internationally]* for his *[brilliant | creative]* insights into *[many | a wide variety of*] *[difficult | challenging*] problems.

He is also a [*highly | greatly*] [*respected | admired*] [*teacher | educator*]. His students give his *[classes | courses] [rave | spectacular]* reviews. He is [*our | the Department's] [mostpopular | bestloved*] [*teacher | instructor*].

[În addition | Additionally] Prof. Wilson is a *[gifted | effective]* fund raiser. His *[grants | contracts]* have brought a *[large | substantial]* amount of money into [*the | our*] Department. *[This money has | These funds have] [enabled | permitted]* us to *[pursue | carry out]* many *[special | important]* programs, [*such as | for example*] your State 2000 program. Without these funds we would [*be enable | not be able*] to continue this program which is so [*important | essential*] to both of us. I strongly urge you to grant him tenure.

Din nefericire pentru Tom, imediat după ce Ellen termină de alcătuit şi introdus mesajul, ea scrie de asemenea un al doilea mesaj:

Dear Dean Smith,

This *[letter | message]* is to give my *[honest | frank]* opinion of Prof. Tom Wilson, who is *[a candidate | up]* for tenure *[now | this year].* I have *[known | worked with].* Prof. Wilson for *[about | almost]* six years. He is an *[poor | weak]* researcher not well known în his *[field | area*]. His research *[hardly ever | rarely]* shows *[insight în | understanding of]* the [*key | major]* problems of [*the | our*] day.

Furthermore, he is not a *[respected | admired] [teacher | educator].* His students give his *[classes | courses] [poor | bad*] reviews. He is [*our | the Department's]* least popular *[teacher | instructor],* known *[mostly | primarily]* within [*our | the*] Department for his *[tendency | propensity]* to *[ridicule | embarrass]* students *[foolish | imprudent]* enough to ask questions în his classes.

[În addition | Additionally] Tom is a *[poor | marginal]* fund raiser. His *[grants | contracts]* have brought only a *[meager | insignificant]* amount of money into [*the | our*] Department. Unless new *[money is | funds are]* quickly located, we must have to cancel essential programs such as your State 2000 program. Unfortunately, under these *[conditions | circumstances].* I cannot în good *[conscience | faith]*

recommend him to you for *[tenure | a permanent position].*

Ellen pune calculatorul să calculeze în timpul nopţii 2^{32} rezumate de mesaj pentru fiecare scrisoare. Există şanse ca un rezumat al primei scrisori să se potrivească cu un rezumat al celei de-a doua scrisori. Dacă nu, ea poate să adauge mai multe opţiuni şi să încerce din nou în timpul weekend- ului. Să presupunem că găseşte o potrivire. Să notăm scrisoarea „bună" cu *A* şi scrisoarea „rea" cu *B.*

Ellen trimite acum scrisoarea *A* lui Marilyn pentru aprobare. Scrisoarea B o păstrează complet secretă, nearătând-o nimănui. Desigur că Marilyn o aprobă, calculează rezumatul de 64 de biţi, semnează rezumatul şi trimite rezumatul semnat decanului Smith. Independent, Ellen îi trimite decanului Smith scrisoarea *B (*şi nu scrisoarea *A*, cum trebuia).

După ce primeşte scrisoarea şi rezumatul semnat, decanul rulează algoritmul de calcul al rezumatului pentru scrisoarea *B*, vede că se potriveşte cu ceea ce i-a trimis Marilyn şi îl dă afară pe Tom. Decanul nu înţelege că Ellen a reuşit să genereze două scrisori cu acelaşi rezumat şi i-a trimis o scrisoare diferită decât cea pe care a văzut-o şi aprobat-o Marilyn. (Sfârşit opţional: Ellen îi povesteşte lui Dick ceea ce a făcut. Dick este revoltat şi rupe relaţia cu ea. Ellen este furioasă şi se confesează lui Marilyn. Marilyn îl sună pe Decan. Tom obţine până la urmă postul). Atacul zilei de naştere este nefezabil la MD5 deoarece, chiar şi la un miliard de rezumate pe secundă, ar trebui peste 500 de ani pentru a calcula 2^{64} rezumate pentru cele două scrisori, cu 64 de variante fiecare, şi nici atunci succesul nefiind garantat. SHA-l este mai bun (deoarece este mai lung).

8.5 GESTIONAREA CHEILOR PUBLICE

Criptografia cu chei publice face posibilă comunicaţia sigură între persoane care nu împart o cheie comună. De asemenea face posibilă semnarea mesajelor fără prezenţa unei a treia părţi de încredere. În fine, rezumatul mesajului semnat face posibilă verificarea uşoară a integrităţii mesajelor primite.

Oricum, există o problemă peste care am trecut un pic cam repede: dacă Alice şi Bob nu se cunosc unul pe altul, cum îşi pot afla cheile publice pentru a porni procesul de comunicaţie? Soluţia evidentă – puneţi cheia publică pe web – nu merge datorită următorului motiv. Presupunem că Alice vrea să caute cheia publică a lui Bob pe site-ul lui de web. Cum face ea acest lucru? Începe prin a tasta adresa de web a lui Bob. Browserul ei va căuta adresa DNS a paginii principale a lui Bob şi va trimite o cerere de tip *GET,* aşa cum se indică în fig. 8-23. Din păcate, Trudy interceptează cererea şi răspunde cu o falsă pagină principală, probabil o copie a paginii principale a lui Bob în care s-a înlocuit cheia publică a lui Bob cu cheia publică a lui Trudy. Când Alice va cripta primul ei mesaj cu E_T, Trudy îl va decripta, îl va citi, îl va recripta cu cheia publică a lui Bob şi îl va trimite apoi lui Bob, care nu este capabil să-şi dea seama că Trudy i-a citit mesajele primite. Şi mai grav, Trudy poate modifica mesajele înainte de a le recripta pentru Bob. În mod evident, este

nevoie de un mecanism care să asigure un schimb sigur de chei publice.

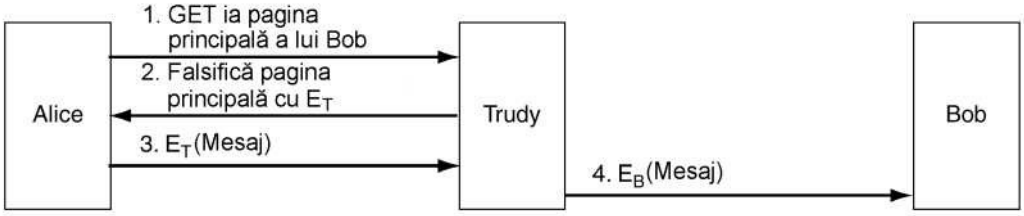


Fig. 8-23. O modalitate prin care Trudy sparge infrastructura cu chei publice

8.5.1 Certificate

Ca o primă încercare de distribuire sigură a cheilor publice, ne putem imagina un centru de distribuţie a cheilor disponibil 24 de ore pe zi pentru a oferi chei publice la cerere. Una dintre multele probleme ale acestei soluţii este lipsa de scalabilitate şi faptul că centrul de distribuţie de chei va deveni rapid un punct de gâtuire a procesului de distribuţie de chei. De asemenea, dacă centrul se va opri vreodată, securitatea Internetului se va pierde în mod subit.

Din cauza acestor motive, s-a dezvoltat o soluţie diferită, care nu presupune un centru de distribuţie de chei care să fie disponibil tot timpul. De fapt, acesta nu trebuie să fie disponibil deloc. În schimb, soluţia aleasă certifică faptul că o cheie publică aparţine unei anume persoane, companii sau altei organizaţii. O organizaţie care certifică chei publice este numită **CA (Certification Authority, rom.: Autoritate de Certificare)**.

Subsemnatul certific faptul că cheia publică
19836A8B03030CF83737E3837837FC3s87092827262643FFA82710382828282A
aparţine lui

Robert John Smith
12345 University Avenue

Berkeley, CA 94702 Birthday: July 4, 1958 Email: bob@superdupernet.com

Rezumatul SHA-l al certificatului de mai sus semnat cu cheia privată a CA

Fig. 8-24. Un certificat posibil şi rezumatul său semnat

De exemplu, să presupunem că Bob vrea să-i permită lui Alice şi altor persoane să comunice în mod sigur cu el. El poate să meargă la un CA cu cheia sa publică însoţită de paşaport sau permisul de conducere şi să ceară să fie certificat. CA-ul va emite un certificat similar cu cel din fig. 8-24 şi va semna rezumatul de tip SHA-l al certificatului cu cheia sa secretă. Bob va plăti preţul cerut de CA şi va obţine o dischetă conţinând certificatul şi rezumatul semnat.Scopul principal al unui certificat este să facă legătura între o cheie publică şi o entitate (individ, companie etc.). Certificatele în sine nu sunt secrete sau protejate. Bob poate, de exemplu, să decidă să-şi pună noul certificat pe sit-ul său web, cu o legătură către pagina sa principală în care sa afirme: "Apasă aici pentru certificatul chei mele publice". Ca rezultat se vor returna atât certificatul cât şi semnătura sa (codul de dispersie sem-

nat SHA-l al certificatului).

Să mai analizam încă o dată scenariul din fig. 8-23. Când Trudy interceptează cererea lui Alice pentru pagina principală a lui Bob, ce poate face ea acum? Îşi poate pune propriul ei certificat şi semnătura pe pagina falsă, dar când Alice va citi certificatul îşi va da seama imediat că nu vorbeşte cu Bob deoarece numele lui Bob nu se găseşte în certificat. Trudy poate modifica pagina lui Bob din mers, înlocuind cheia publică a lui Bob cu propria ei cheie. Oricum, când Alice va rula algoritmul SHA-l pe certificat, va obţine un rezumat care va fi diferit de cel obţinut prin aplicarea cheii publice a CA-ului asupra semnăturii certificatului. Cum Trudy nu deţine cheia privată a CA-ului, ea nu are nici o modalitate de a genera un bloc de semnătură care să conţină codul de dispersie modificat de ea, prin înlocuirea propriei sale chei publice în pagina de web. În acest fel Alice poate fi sigură ca are cheia publică a lui Bob şi nu pe cea a lui Trudy sau a altcuiva. Cum am promis, această schemă nu presupune un CA disponibil mereu pentru verificare, eliminându-se astfel o posibilă gâtuire.

În timp ce funcţia standard a unui certificat este de a face legătura între o cheie publică şi o entitate, un certificat poate face legătura între o cheie publică şi un atribut. De exemplu un certificat poate afirma: "Această cheie publică aparţine cuiva care are vârsta de peste 18 ani. Ea poate fi folosită pentru a dovedi că posesorul cheii private nu este minor şi că are acces la materiale interzise copiilor şi aşa mai departe" , fără a dezvălui identitatea posesorului certificatului. În mod obişnuit, persoana care deţine un certificat, îl va trimite unui sit Web, persoană sau proces de administrare, în care se ţine cont de vârstă. Acel sit (persoana sau procesul de administrare) va genera un număr aleator şi îl va cripta cu cheia publică prezentă în certificat. Dacă posesorul certificatului este capabil să decripteze mesajul şi să-l trimită înapoi, acest lucru va fi o dovada că posesorul deţine într-adevăr atributul din certificat. Ca alternativă, numărul aleator poate fi folosit pentru a genera o cheie de sesiune pentru a garanta comunicaţia.

Un alt exemplu în care certificatul poate conţine un atribut este cazul sistemelor distribuite orientate pe obiecte. Fiecare obiect are în mod normal mai multe metode. Proprietarul unui obiect poate oferi fiecărui client un certificat care să conţină o hartă de biţi cu metode ce sunt permise clientului respectiv şi poate face legătura între cheia publică şi harta de biţi prin folosirea unui certificat semnat. Şi în acest caz, dacă posesorul certificatului poate dovedi că este în posesia cheii secrete corespunzătoare, el va avea dreptul să execute metodele identificate de harta de biţi. Certificatul are proprietatea că identitatea posesorului nu trebuie să fie cunoscută, proprietate utilă în situaţii în care confidenţialitatea este importantă.

8.5.2 X.509

Dacă fiecare persoană care doreşte ceva semnat ar merge la CA pentru diverse tipuri de certificate, gestionarea tuturor tipurilor de formate ar deveni curând o problemă. Pentru a rezolva această

problemă, s-a proiectat şi aprobat de către ITU un standard pentru certificate. Standardul se numeşte X.509 şi este folosit pe scara largă în Internet. De la prima

standardizare din 1998, au existat trei versiuni. Vom discuta mai departe despre V3.

X.509 a fost foarte mult influenţat de lumea OSI, împrumutând unele din cele mai proaste trăsături (ex. politica de nume şi codificarea). În mod surprinzător, IETF a fost de acord cu X.509, chiar dacă în alte domenii, de la adresele maşinilor la protocoalele de transport şi formatul poştei electronice, IETF ignoră OSI şi încearcă să facă lucrurile corect. Versiunea IETF pentru X.509 este descrisă în RFC 3280.

În principal, X.509 este o modalitate de a descrie certificate. Câmpurile principale dintr-un certificat sunt listate în fig. 8-25. Descrierea dată aici ar trebui să ofere o idee generală a ceea ce fac câmpurile respective. Pentru informaţii adiţionale, vă rog să consultaţi standardul în sine sau RFC 2459.

Câmp	**Semnificaţie**
Versiune	Ce versiune de X.509 este utilizată
Număr Serial	Acest număr împreună numele CA-ului identifică în mod unic
Algoritm de semnare	Algoritmul folosit la semnarea certificatului
Emitent	Numele X.500 al CA-ului
Perioada de	Momentele de început şi sfârşit ale perioadei de validitate
Numele subiectului	Entitatea care este certificată
Cheia publică	Cheia publică a subiectului şi ID-ul algoritmului folosit
ID emitent	Un identificator opţional identificând în mod unic emitentul
ID subiect	Un identificator opţional identificând în mod unic subiectul
Extinderi	Au fost definite mai multe extinderi
Semnătura	Semnătura certificatului (semnat cu cheia privată a CA-ului)

Fig. 8-25. Câmpurile principale dintr-un certificat X.509

De exemplu, daca Bob lucrează în departamentul de împrumuturi al Băncii Bani, adresa sa X.500 poate sa fie:

/c=US/O=MoneyBanc/OU=Loan/CN=Bob/

unde C indică ţara, O indica organizaţia, OU reprezintă o unitate din organizaţie, şi CN este folosit drept numele comun (eng.: common name). CA-urile şi alte entităţi sunt denumite similar. O problemă mare cu numele X.500 este că, dacă Alice vrea să-l contacteze pe bob@moneybank.com şi are un certificat conţinând un nume X.500, nu este evident că acel certificat se referă la acel Bob pe care vrea ea să-l contacteze. Din fericire, începând cu versiunea 3, sunt permise numele DNS în loc de numele X.500, deci această problemă ar putea să dispară.

Certificatele sunt codificate folosind **OSI ASN.1 (eng.: Abstract Syntax Notation 1, rom.: Notaţia sintactică abstractă 1)**, care poate fi văzută ca o structură C, cu excepţia unei notaţii foarte specifice şi detaliate. Mai multe informaţii despre X.509 pot fi găsite în (Ford şi Baum, 2000).

8.5.3 Infrastructuri cu chei publice

Existenţa unei singure Autorităţi de certificare care să emită toate certificatele din lume nu este evident o bună soluţie. Ea ar ceda datorită încărcării mari şi va fi în acelaşi timp şi un punct central de defectare. O soluţie posibilă este existenţa mai multor CA-uri, toate rulând în cadrul aceleaşi organizaţii şi folosind aceeaşi cheie privată pentru semnarea certificatelor. Cu toate că se rezolvă problema încărcării şi

a defectării, această soluţie introduce o problemă nouă: dezvăluirea cheilor. Dacă ar exista o zeci de servere împrăştiate prin lume, toate deţinând cheia privată a CA-ului, posibilitatea furtului cheii private sau a dezvăluirii ei va creşte foarte mult. Cum compromiterea acestei chei va ruina securitatea infrastructurii electronice mondiale, existenţa unei singure CA centrale este un risc foarte mare.

În plus, ce organizaţii vor juca rolul de CA? Este foarte greu de imaginat că orice autoritate este acceptată la nivel mondial ca legitimă şi de încredere. În unele ţări oamenii vor insista să fie guvernamentală, în timp ce în altele vor insista să nu fie o organizaţie guvernamentală.

Din cauza acestor motive, a fost dezvoltată o altă variantă de certificare a cheilor publice. Denumirea generală este de **PKI (**eng.: **Public Key Infrastructura**, rom.: **Infrastructură cu chei publice)** . În această secţiune vom prezenta pe scurt cum funcţionează PKI în general, deşi au existat mai multe propuneri care vor face ca detaliile să evolueze în timp.

O PKI are mai multe componente, incluzând utilizatorii, CA-urile, certificatele şi directoarele. Scopul unei PKI este să ofere o structurare a acestor componente şi să definească standarde pentru diferite documente şi protocoale. O formă particulară de PKI este o ierarhie de CA-uri, aşa cum se arată în fig. 8-26. În acest exemplu se prezintă trei niveluri, dar în practică pot fi mai multe sau mai puţine. CA-ul din vârf, rădăcina, certifică a doua CA, pe care o vom denumi **RA (**eng.: **Regional Authority**, rom.: **Autoritate Regională)** deoarece va răspunde de anumite regiuni geografice, precum o ţară sau un continent. Acest termen nu este standard; de fapt nici un termen referitor la diferite nivele din arbore nu este standardizat. Aceste RA-uri certifică de fapt adevăratele CA-uri, care vor emite certificate X.509 pentru organizaţii şi indivizi. Când rădăcina autorizează o nouă RA, ea va genera un nou certificat X.509 care atestă că a aprobat RA-ul, incluzând în el noua cheie publică a RA-ului, îl va semna şi îl va trimite RA-ului. Similar, RA-ul aprobă noi CA-uri, generează şi semnează certificate care atestă aprobarea şi conţin cheia publică a CA-ului.

PKI-ul nostru funcţionează în modul următor. Să presupunem că Alice are nevoie de cheia publică a lui Bob pentru a comunica cu acesta, deci ea va căuta şi va găsi un certificat semnat de CA 5 care să conţină cheia publică respectivă. Dar Alice nu a auzit niciodată de CA 5. Din câte ştie ea, CA 5 poate fi şi fiica de 10 ani a lui Bob. S-ar putea duce la CA5 şi să-i pretindă să îşi dovedească legitimitatea. CA 5 răspunde cu certificatul obţinut de la RA 2, care conţine cheia publică a lui CA 5. Acum, deţinând cheia publică a lui CA 5, ea poate verifica dacă certificatul lui Bob este într-adevăr semnat de CA 5 şi datorită acestui fapt este legal.

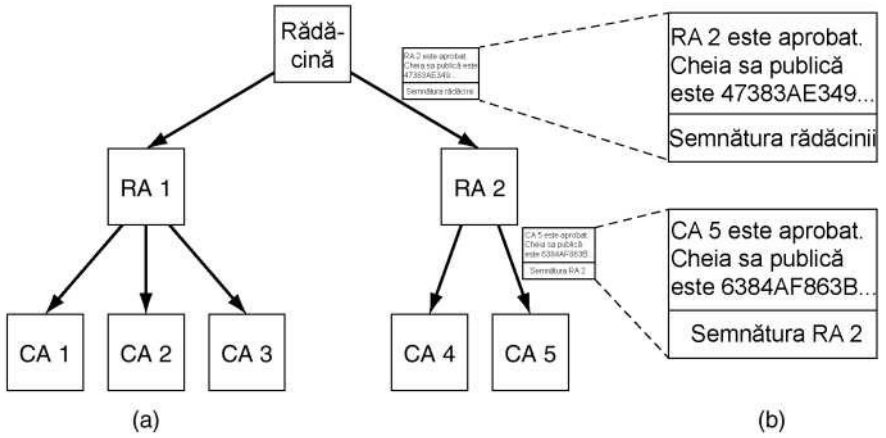


Fig. 8-26. (a) O PKI ierarhică (b) Un lanţ de certificate

Dar dacă RA 2 este copilul de 12 ani al lui Bob? Deci pasul următor pentru ea este de a cere ca RA 2 să-şi dovedească legitimitatea. Răspunsul la întrebarea ei este certificatul semnat de rădăcină şi care conţine cheia publică a lui RA 2. Acum Alice este sigură ca ea deţine cheia publică a lui Bob.

Dar cum ştie Alice sa găsească cheia publică a rădăcinii? Magie. Se presupune că toată lumea cunoaşte cheia publică a rădăcinii. De exemplu, programul său de navigare ar fi putut fi vândut cu cheia publică a rădăcinii înglobată în el.

Bob este un tip prietenos şi nu vrea să-i provoace lui Alice multă bătaie de cap. El ştie că ea trebuie să verifice CA 5 şi RA 2, deci pentru a-i economisi timp, el colectează certificatele necesare şi îi transmite încă două certificate alături de al său. Acum ea îşi poate folosi propria ei informaţie cu privire la cheia publică a rădăcinii pentru a verifica certificatul din vârful ierarhiei şi cheia publică conţinută în acesta pentru a verifica al doilea certificat. În acest fel, Alice nu mai trebuie să contacteze pe nimeni pentru a efectua verificarea. Pentru că toate certificatele sunt semnate, ea poate detecta cu uşurinţa orice încercare de falsificare a conţinutului acestora. Un lanţ de certificate care duce la rădăcină, ca în acest exemplu, este denumit câteodată **lanţ de încredere** sau **cale de certificare.** Tehnica este larg răspândită în practică.

Desigur, mai rămâne problema cine va juca rolul de rădăcină. Soluţia este să nu avem o singură rădăcină, ci să avem mai multe, fiecare cu propriile sale RA-uri şi CA-uri. De fapt, programele de navigare moderne vin preîncărcate cu cheile publice a peste 100 de rădăcini, denumite câteodată **puncte de încredere**. În acest fel, poate fi evitată existenţa la nivel mondial a unei singure autorităţi de încredere.

Dar apare acum problema cum poate decide producătorul programului de navigare care dintre punctele sugerate sunt de încredere şi care nu. Totul se reduce la încrederea utilizatorului în faptul că producătorul programului de navigare este capabil să facă alegeri înţelepte şi nu doar să aprobe toate punctele de încredere care au plătit o taxă pentru includere. Majoritatea programelor de navigare permit utilizatorilor să inspecteze cheile rădăcinilor (în mod obişnuit sub formă de certificate semnate de rădăcină) şi să le şteargă pe cele care par dubioase.

Directoare

O altă problemă pentru orice PKI este locul în care se stochează certificatele (şi lanţurile care duc către un punct de încredere cunoscut). O posibilitate este ca fiecare utilizator să-şi stocheze propriile certificate. Cu toate că acest lucru este sigur (de ex. nu există nici o posibilitate ca un utilizator să falsifice certificatele semnate fără a fi detectat), nu este prea convenabil. O alternativă care a fost propusă este folosirea DNS ca director pentru certificate. Înainte de a-l contacta pe Bob, Alice probabil îi va căuta adresa IP folosind DNS, deci de ce să nu întoarcă DNS-ul întregul lanţ de certificate odată cu adresa IP?

Unele persoane consideră că aceasta este soluţia corectă, dar alţii preferă servere de directoare specializate a căror îndatorire este doar gestionarea de certificate X.509. Directoarele de acest fel pot oferi servicii de căutare bazate pe proprietăţile numelor X.500. De exemplu, în teorie un astfel de director poate răspunde la o întrebare de tipul: "Dă-mi o listă a persoanelor numite Alice care lucrează în departamentul de vânzări oriunde în Canada sau S.U.A.". LDAP poate fi un candidat care să stocheze acest tip de informaţii.

Revocarea

Lumea reală este plină de certificate, la fel ca în cazul paşapoartelor şi al carnetelor de conducere. Câteodată aceste certificate pot fi revocate, de exemplu, carnetul de conducere poate fi suspendat pentru conducere sub influenţa alcoolului sau pentru o altă greşeală de conducere. Aceeaşi problemă apare şi în lumea digitală: emitentul unui certificat poate decide dacă să-l revoce problemă apare şi în lumea digitală: emitentul unui certificat poate decide dacă să-l revoce în cazul în care persoana sau organizaţia care îl deţine a abuzat de el într-un anume mod. El poate fi de asemenea revocat dacă cheia privată a subiectului a fost compromisă (demascată), sau mai rău, dacă cheia privată a CA-ului a fost compromisă. Atunci, o PKI trebuie să fie capabilă să trateze problema revocării.

Primul pas în această direcţie este ca fiecare CA să emită periodic o **CRL** (eng.: **Certificate Revocation List,** rom.: **Listă de Certificate Revocate**) conţinând numerele seriale ale tuturor certificatelor revocate de aceasta. Cum certificatele conţin data de expirare, CRL-ul trebuie să conţină doar numerele seriale ale certificatelor care încă nu au expirat. Odată ce perioada de validitate a expirat, un certificat este automat invalidat, deci nu mai este necesară nici o distincţie între acelea care tocmai au expirat şi cele are au fost de fapt revocate. În ambele cazuri, certificatele nu mai pot fi folosite în continuare.

Din păcate, introducerea CRL-ului înseamnă că un utilizator care doreşte să folosească un certificat trebuie acum să consulte CRL-ul pentru a vedea dacă certificatul a fost revocat. Dacă a fost revocat, nu mai trebuie folosit. Oricum, chiar dacă certificatul nu este în listă, el poate să fi fost revocat chiar după ce lista a fost publicată. Atunci, singura modalitate rămâne întrebarea către CA. La o nouă folosire a aceluiaşi certificat, CA-ul trebuie întrebat din nou, pentru este posibil ca certificatul să fi fost revocat cu câteva secunde în urmă.

O altă complicaţie este că un certificat revocat poate în principiu să fie restabilit, de exemplu, dacă a fost revocat datorită neplăţii unei taxe care acum s-a plătit. Având de a face cu revocarea (şi posibil cu restabilirea) se elimină una dintre cele mai bune proprietăţi ale certificatelor, adică faptul că ele pot fi folosite fără a fi necesar sa se contacteze o RA.

Unde trebuie stocate CRL-urile? Un loc bun ar putea fi acela în care se stochează certificatele însele. O strategie este ca CA-ul să publice periodic CRL-uri şi să permită directoarelor să înlăture certificatele revocate. Dacă nu sunt folosite directoare pentru stocarea de certificate, CRL-urile pot fi ţinute în diverse locuri convenabile din reţea. Cum o CRL este ea însăşi un document, dacă este falsificată, falsul se poate detecta uşor.

Dacă certificatele au durate de viaţă lungi, şi CRL-urile vor avea perioade lungi. De exemplu, când cărţile de credit sunt valide pentru 5 ani, numărul de nerezolvări ale revocărilor va fi mult mai mare decât dacă se emit cărţi de credit noi la fiecare 3 luni. Un mod standard de a aborda CRL-urile lungi este de a emite rar o listă principală, dar de a-i face modificări mult mai des. Acest lucru reduce lăţimea de bandă necesară distribuirii CRL-urilor.

8.6 SECURITATEA COMUNICAŢIEI

Am terminat acum studiul asupra instrumentelor comerciale. Cele mai importante tehnologii şi protocoale au fost prezentate. Restul capitolului se referă la modul de aplicare în practică a acestor tehnologii pentru a se asigura securitatea reţelei, plus câteva gânduri despre aspectele sociale ale securităţii prezentate la sfârşitul capitolului.

În următoarele patru secţiuni, ne vom axa pe securitatea comunicaţiei, deci cum să transferăm biţii confidenţial şi fără a fi modificaţi, de la sursă la destinaţie şi cum să ţinem biţii nedoriţi în faţa

uşii. Fără discuţie că acestea nu sunt singurele idei despre securitatea în reţele, dar cu siguranţă sunt printre cele mai importante, constituind un bun loc de plecare.

8.6.1 IPsec

IETF a ştiut de ani de zile că securitatea lipsea din Internet. Adăugarea ei nu a fost uşoară deoarece a izbucnit un război cu privire la locul de plasare al acesteia. Majoritatea experţilor în securitate credeau că pentru a fi într-adevăr sigure, criptarea şi verificarea integrităţii trebuie să fie capăt la capăt (de ex. nivelul aplicaţie). Adică, procesul sursă criptează şi/sau protejează datele din punct de vedere al integrităţii şi le trimite către procesul de destinaţie, unde sunt decriptate şi/sau verificate. Orice falsificare între aceste două procese, incluzând sistemele lor de operare, poate fi detectată. Problema cu această abordare este că presupune schimbarea tuturor aplicaţiilor pentru a le securiza. În această idee, următoarea abordare posibilă este plasarea criptării pe nivelul transport sau într-un nou nivel între nivelul aplicaţie şi nivelul transport, păstrând mecanismul tot capăt la capăt, dar fără a mai fi necesare schimbări ale aplicaţiilor.

Ideea opusă este că utilizatorii nu înţeleg securitatea şi că nu vor fi capabili să o folosească în mod corect şi nimeni nu doreşte să modifice programele existente în nici un fel, deci nivelul reţea trebuie să autentifice şi/sau cripteze pachetele fără

nici o implicare din partea utilizatorilor. După ani de controverse, această idee a câştigat suficient suport astfel încât a fost definit un standard de securitate pentru nivelul reţea. În parte, argumentul a fost că având criptarea la nivelul reţea nu împiedică utilizatorii conştienţi de problema securităţii să o folosească corect şi îi ajută într-un anumit grad pe utilizatorii neavizaţi.

Rezultatul acestui război a fost un proiect numit IPsec (IP securizat), care este descris în RFC- urile 2401, 2402 şi 2406, printre altele. Nu toţi utilizatorii doresc criptarea (pentru că este costisitoare din punct de vedere al puterii de calcul). În loc să fie opţională, s-a decis să se impună criptarea permanent, dar cu posibilitatea de folosire unui algoritm nul. Algoritmul nul este descris şi lăudat pentru simplitatea sa, uşurinţa de implementare şi viteza mare în RFC 2410.

Proiectul complet IPsec este o cadru de lucru pentru mai multe servicii, algoritmi şi granularităţi. Motivul pentru servicii multiple este că nu toată lumea doreşte să plătească preţul necesar tuturor serviciilor, deci serviciile sunt disponibile la cerere. Serviciile principale sunt confidenţialitatea, integritatea datelor, şi protejarea lor de atacul prin replicare (un intrus poate replica un dialog). Toate acestea se bazează pe criptografia cu chei simetrice pentru că înalta performanţă este un lucru crucial.

Motivul pentru care există mai mulţi algoritmi este că un algoritm care acum pare sigur poate fi spart în viitor. Făcând IPsec-ul independent de algoritmi, cadrul de lucru poate supravieţui chiar dacă mai târziu un algoritm este spart.

Motivul pentru care există mai multe granularităţi se datorează posibilităţii de a se proteja o singură conexiune TCP, tot traficul dintre două gazde, sau tot traficul dintre o pereche de rutere securizate, printre alte posibilităţi.

Un aspect destul de surprinzător al IPsec-ului este că deşi se găseşte pe nivel IP, el este orientat pe conexiune. De fapt, nu este chiar aşa de surprinzător deoarece pentru a exista securitate, o cheie trebuie să fie stabilită şi folosită pentru o anume perioadă de timp, în esenţă, un fel de conexiune. De asemenea conexiunile amortizează costurile de iniţializare pentru mai multe pachete. În contextul IPsec o "conexiune" este denumită SA (eng.: Security Association, rom.: asociere securizată). O SA este o conexiune simplă între două capete şi are asociat un identificator de securitate.

Dacă este nevoie de un trafic securizat în ambele direcţii, sunt necesare două asocieri securizate. Identificatorii de securitate sunt transportaţi în pachetele care se transmit pe aceste conexiuni securizate şi sunt folosiţi la căutarea cheilor sau a altor informaţii relevante atunci când soseşte un pachet securizat.

Din punct de vedere tehnic, IPsec conţine două părţi principale. Prima parte descrie două noi antete care pot fi adăugate pachetelor pentru a transporta identificatorul de securitate, datele de control al integrităţii şi alte informaţii. Cealaltă parte, **ISAKMP** (eng.: **Internet Security Association and Key Managment Protocol**, rom.: **Asociaţia Securităţii Internet şi Protocolul de Gestiune al Cheilor**) se ocupă cu stabilirea cheilor. Nu vom discuta mai departe despre ISAKMP deoarece (1) este foarte complex şi (2) protocolul său principal **IKE** (eng.: **Internet Key Exchange**, rom.: **Schimbul de Chei Internet**), este foarte deficient şi trebuie să fie înlocuit (Perlman şi Kaufman, 2000).

IPsec poate fi folosit în două moduri. În modul transport, antetul IPsec este inserat chiar după antetul IP. Câmpul de *Protocol* din antetul IP este schimbat pentru a indica faptul că un antet IPsec urmează după antetul normal de IP (înainte de antetul TCP). Antetul IPsec conţine informaţii de securitate, în principal identificatorul SA, un nou număr de secvenţă şi, posibil, o verificare a integrităţii încărcăturii utile.

În modul tunel, întregul pachet IP, antet şi restul, este încapsulat în corpul unui nou pachet IP cu un antet IP complet nou. Modul tunel este folositor când capetele tunelului se termină la o locaţie diferită de destinaţia finală. În unele cazuri, capătul tunelului este o maşină poartă de aplicaţie (eng.: gateway) securizată, de exemplu, de zidul de protecţie (eng.: firewall) al unei companii. În acest mod, zidul de protecţie încapsulează şi decapsulează pachete la trecerea prin el. Terminând tunelul prin această maşină securizată, maşinile din LAN-ul companiei nu trebuie să fie conştiente de IPsec. Doar zidul de protecţie trebuie să ştie de el.

Modul tunel este de asemenea util când o legătura a unei conexiuni TCP este agregată şi se comportă ca un flux criptat, deoarece împiedică un intrus să vadă câte pachete trimite cineva către altcineva. Câteodată doar cunoscând cât de mult trafic este dirijat undeva reprezintă o informaţie de valoare. De exemplu, dacă în timpul unei crize militare, dimensiunea traficului dintre Pentagon şi Casa Albă scade drastic, dar cantitatea de trafic dintre Pentagon şi un obiectiv militar aflat în creierul munţilor Stâncoşi ai Colorado-ului creşte în aceeaşi măsură un intrus e capabil să deducă câteva informaţii utile din aceste date. Studierea şabloanelor de scurgere a pachetelor, chiar dacă ele sunt criptate, se numeşte **analiză a traficului**. Modul tunel oferă o modalitate de zădărnicire într-o anume măsură. Dezavantajul modului tunel este că adaugă un antet IP în plus, crescând astfel în mod substanţial dimensiunea unui pachet. Prin contrast, modul transport nu afectează atât de mult dimensiunea pachetului.

Primul antet nou este **AH (eng.: Authentication Header, rom.: Antetul de Autentificare)**. El permite controlul integrităţii şi securitate anti-replică, dar nu şi confidenţialitate (de ex. nu criptează datele). Folosirea AH în modul tunel este ilustrată în fig. 8-27. În IPv4, el este interpus între antetul IP (incluzând orice opţiuni) şi antetul TCP. În IPv6 este doar o altă extensie a antetului şi este tratată ca atare. De fapt, formatul este apropiat de cel al standardului IPv6 de extensie a antetului. Este posibil ca încărcătura utilă să trebuiască a fie completată până la o anumită lungime pentru algoritmul de autentificare după cum este prezentat în figură.

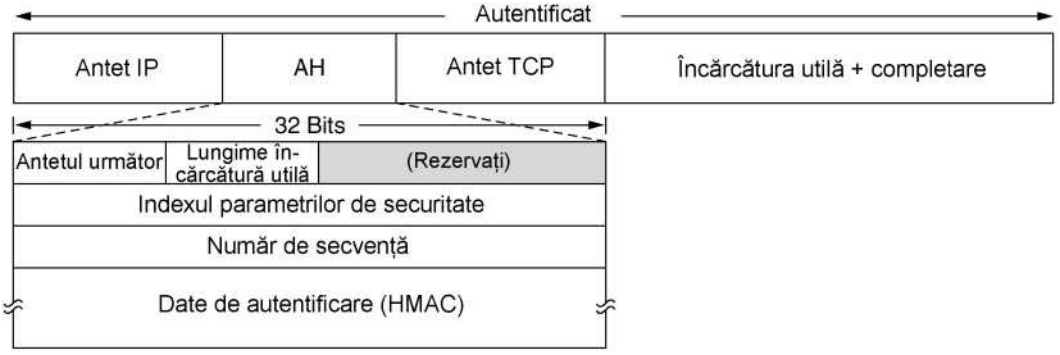


Fig. 8-27. Antetul de autentificare IPsec în modul transport pentru IPv4

Să examinăm acum antetul AH. Câmpul *Antetul următor* este folosit pentru a păstra valoarea anterioară pe care a avut-o câmpul *Protocol* IP înainte de a fi înlocuit cu 51 pentru a indica faptul că urmează un antet AH. În majoritatea cazurilor, aici va fi plasat codul pentru TCP (6) . *Lungimea încărcării utile* o reprezintă numărul de cuvinte de 32 de biţi din antetul AH minus 2.

Indexul parametrilor de securitate reprezintă identificatorul de conexiune. Acesta este inserat de către emiţător pentru a indica o anumită înregistrare în baza de date a receptorului. Această înregistrare conţine cheia partajată folosită în această sesiune şi alte informaţii despre conexiune. Dacă acest protocol ar fi fost inventat de către ITU şi nu de către IETF, acest câmp ar fi fost denumit *Număr de circuit virtual*.

Câmpul *Număr de secvenţă* este folosit pentru a număra toate pachetele trimise pe un SA. Fiecare pachet primeşte câte un identificator unic, chiar şi retransmisiile, cu alte cuvinte copia unui pachet primeşte un număr diferit de cel original (chiar dacă numărul său de secvenţă TCP este acelaşi). Scopul acestui câmp este de a detecta atacurile prin replică. Aceste numere de secvenţă nu se pot repeta. Dacă toate cele 2^{32} de numere au fost epuizate, trebuie stabilit un nou SA pentru a continua comunicaţia.

În sfârşit, câmpul *Date de autentificare* este un câmp de lungime variabilă care conţine semnătura digitală. Când este stabilit un SA, cele două părţi negociază algoritmul de semnare pe care îl vor folosi. În mod normal, nu este folosită criptografia cu chei publice pentru că pachetele trebuie procesate rapid, iar toţi algoritmii cu chei publice sunt prea lenţi. Deoarece IPsec este bazat pe criptografia cu chei simetrice iar emiţătorul şi receptorul negociază o cheie partajată înaintea stabilirii unui SA, cheia partajată este folosită în procesul de semnare. O modalitate simplă este de a calcula rezumatul pentru un pachet şi cu cheia partajată . Desigur, cheia partajată nu este transmisă . O schemă ca aceasta este denumită **HMAC (eng. : Hashed Message Authentication Code, rom.: Cod de Autentificare bazat pe un rezumat de mesaj)** . Este mult mai rapid să calculezi un rezumat decât să rulezi întâi SHA-l şi apoi să rulezi RSA asupra rezultatului.

Antetul AH nu permite criptarea datelor, deci este cel mai folosit atunci când este necesară verificarea integrităţii dar nu este necesară confidenţialitatea. O proprietate demnă de notat a antetului AH este aceea că verificarea integrităţii

foloseşte o parte din câmpurile antetului IP, şi anume, acelea care nu se schimbă când pachetul trece de la un ruter la altul. De exemplu, câmpul *Durata de viaţă* se schimbă la fiecare ruter şi nu poate fi inclus în verificarea integrităţii. În orice caz adresa sursa IP este inclusă în această verificare, un intrus neputând să falsifice originea pachetului.

O altă variantă de antet IPsec este **ESP (eng.: Encapsulating Security Payload, rom.: Încapsularea încărcăturii utile de securitate).** Folosirea acestuia, atât pentru modul transport, cât şi pentru modul tunel, este prezentată în fig. 8-28.

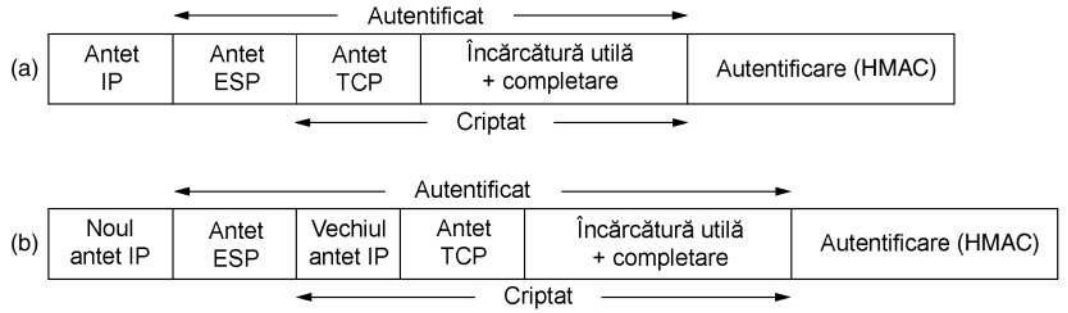


Fig. 8-28. (a) ESP în mod transport (b) ESP în mod tunel

Antetul ESP conţine două cuvinte de 32 de biţi. Acestea sunt câmpurile *Indexul Parametrilor de Securitate* şi, respectiv, *Numărul de Secvenţă* pe care le-am întâlnit la AH. Un al treilea cuvânt care, în general, le urmează (dar care din punct de vedere tehnic nu este parte a antetului) este câmpul *Vector de Iniţializare,* folosit pentru criptarea datelor; dacă nu se foloseşte criptarea acest câmp este omis.

De asemenea, ESP oferă verificarea integrităţii pentru HMAC, aşa cum o face şi AH, dar în loc să fie inclus în antet, urmează după încărcătura utilă, aşa cum se arată în fig. 8-28. Includerea HMAC la sfârşit are un avantaj în implementarea hardware. HMAC poate fi calculat pe măsură ce biţii ies dintr-o interfaţă de reţea ataşarea sa la sfârşit fiind foarte simplă . Din această cauză Ethernet şi alte LAN-uri au propriile CRC la sfârşitul pachetelor în loc să le aibă la începutul acestora. Cu AH, pachetul trebuie să fie păstrat într-o zonă tampon şi semnătura să fie calculată înainte ca pachetul să fie trimis, reducând astfel numărul de pachete care pot fi transmise pe secundă.

Deoarece ESP poate face tot ceea ce poate face AH şi chiar mai mult şi faptul că este mai eficient la iniţializare decât acesta, se ridică întrebarea: De ce să ne mai complicăm cu AH? Răspunsul este mai mult istoric. La început, AH se ocupa numai de integritate iar ESP numai de confidenţialitate. Mai târziu, integritatea a fost adăugată şi la ESP, dar cei care au proiectat AH nu au vrut ca acesta să dispară după ce au muncit atât de mult la el. În orice caz, singurul lor argument real este faptul că AH verifică o parte a antetului IP, ceea ce ESP nu face; dar acesta este un argument nesemnificativ. Un alt argument nesemnificativ este acela că un produs care suportă AH, dar nu suportă ESP, poate avea mai puţine probleme în a obţine licenţa de export, datorită faptului că nu foloseşte criptarea. Este posibil ca în viitor AH să nu mai fie folosit.

8.6.2 Ziduri de protecţie

Posibilitatea de a conecta orice calculator, de oriunde, cu orice alt calculator, de oriunde, este o sabie cu două tăişuri. Pentru persoanele aflate acasă, colindatul prin Internet aduce multe bucurii. Pentru administratorii pe probleme de securitate ai firmelor, este un coşmar. Multe companii au mari cantităţi de informaţie confidenţială sub formă electronică – secrete de afaceri, planuri de dezvoltare produse, strategii de marketing, analize financiare etc. Dezvăluirea acestor informaţii către un competitor poate avea consecinţe cumplite.

În afara pericolului scurgerii de informaţii, există şi un pericol al infiltrării de informaţii. În particular, viruşii, viermii şi alţi dăunători digitali pot încălca securitatea, distruge informaţii de valoare şi irosi o mare cantitate din timpul administratorilor care încearcă să cureţe dezordinea pe care o lasă. Deseori ei sunt importaţi de angajaţi neglijenţi care vor să joace un joc nou, grozav.

În consecinţă, sunt necesare mecanisme pentru a păstra biţii „buni” în interior şi biţii „răi” afară. O metodă este folosirea IPsec. Această abordare protejează datele în tranzit între situri sigure. Cu toate acestea, IPsec nu face nimic pentru a ţine dăunătorii digitali şi spărgătorii în afara LAN-ului companiei. Pentru a realiza acest obiectiv, este nevoie să studiem zidurile de protecţie (eng.: firewalls).

Fig. 8-29. Un zid de protecţie format din două filtre de pachete şi o poartă de

Zidurile de protecţie sunt doar o adaptare modernă a acelei vechi soluţii de securitate medievală: săparea unui şanţ adânc de apărare în jurul castelului dvs. Acest proiect forţa pe oricine dorea să intre sau să iasă din castel să treacă peste un singur pod mobil, unde puteau fi inspectaţi de poliţia de intrare/ieşire. Cu reţelele se poate face acelaşi truc: o companie poate avea multe LAN-uri conectate în moduri arbitrare, dar tot traficul către sau de la companie este forţat printr-un pod mobil electronic (zidul de protecţie), aşa cum este prezentat în fig. 8-29.

În această configuraţie zidul de protecţie are două componente: două rutere care

fac filtrare de pachete şi o poartă de aplicaţii. Există, de asemenea, configuraţii şi mai simple, dar avantajul acestui proiect este că fiecare pachet trebuie să tranziteze două filtre şi o poartă de aplicaţie pentru a intra sau ieşi. Nu există altă rută. Este destul de clar că cititorii care consideră că un singur punct de verificare a securităţii este suficient nu au făcut recent un zbor internaţional cu o companie aeriană.

Fiecare **filtru de pachete** este un ruter standard echipat cu unele funcţii suplimentare. Acestea permit inspectarea fiecărui pachet care intră sau iese. Pachetele care îndeplinesc anumite criterii sunt transmise normal. Cele care nu trec testul sunt eliminate.

În fig. 8-29, este foarte probabil ca filtrul de pachete din interiorul LAN-ului să verifice pachetele care ies, iar cel din exteriorul LAN-ului să verifice pachetele care intră. Pachetele care trec de prima barieră merg la poarta de aplicaţii pentru o examinare ulterioară. Motivul introducerii a două filtre de pachete în reţele diferite este de a asigura că nici un pachet nu intră sau iese fără a fi obligat să treacă prin poarta de aplicaţii: nu există nici o cale pe care să o ocolească.

Filtrele de pachete sunt, în mod tipic, dirijate de tabele configurate de administratorul de sistem. Aceste tabele enumeră sursele şi destinaţiile acceptabile, sursele şi destinaţiile care sunt blocate şi reguli implicite despre ce se face cu pachetele care vin sau se duc la alte maşini.

În cazul uzual al configurării TCP/IP, o sursă şi o destinaţie constau dintr-o adresă IP şi un port. Porturile indică ce serviciu este dorit. De exemplu, portul 23 TCP este pentru Telnet, portul 79 TCP este pentru Finger, iar portul 119 TCP este pentru ştirile USENET. O companie poate bloca toate pachetele de intrare pentru toate adresele IP asociate cu unul din aceste porturi. În acest mod, nimeni din afara companiei nu se poate conecta prin telnet, sau să caute persoane folosind demonul de Finger. Mai mult, compania va fi scutită ca angajaţii să-şi petreacă toată ziua citind ştiri USENET.

Blocarea pachetelor care ies este mai complicată, deoarece, deşi cele mai multe situri aderă la convenţiile standard de numire a porturilor, nu sunt obligate să o facă. Mai mult, pentru servicii importante, cum ar fi FTP (File Transfer Protocol – protocol de transfer fişiere), numerele de port sunt atribuite dinamic. În plus, deşi blocarea conexiunilor TCP este dificilă, blocarea pachetelor UDP este şi mai grea datorită faptului că se ştie foarte puţin a priori despre ce vor face. Multe filtre de pachete pur şi simplu interzic în totalitate traficul UDP.

A doua jumătate a mecanismului de zid de protecţie este **poarta de aplicaţie**. În loc să trateze pachete brute, o poartă operează la nivelul aplicaţie. O poartă de poştă electronică, de exemplu, poate fi configurată să examineze fiecare mesaj care intră sau iese. Pentru fiecare mesaj, ea ia decizia de a-l transmite sau elimina pe baza câmpurilor din antet, a dimensiunii mesajului sau chiar a conţinutului (de exemplu, la o instalaţie militară, prezenţa cuvintelor ca „nuclear” sau „bombă” pot cauza generarea unor acţiuni speciale).

Proiectele de instalare au libertatea de a configura una sau mai multe porţi de aplicaţie pentru aplicaţii specifice, dar nu este ieşit din comun ca organizaţii suspicioase să permită intrarea şi ieşirea poştei electronice şi, probabil, folosirea World Wide Web, dar să interzică orice altceva ca fiind prea riscant. Combinat cu

criptarea şi cu filtrarea de pachete, acest aranjament oferă o cantitate limitată de securitate cu costul unor inconveniente.

Chiar dacă zidul de protecţie este configurat perfect, există încă o groază de probleme de securitate. De exemplu, dacă un zid de protecţie este configurat să accepte pachete doar de la anumite reţele (de ex. alte sedii ale companiei), un intrus din exteriorul zidului de protecţie poate să-şi pună o adresă falsă pentru a evita această verificare. Dacă un individ din interior doreşte să vândă documentele secrete, el le poate cripta sau chiar fotografia şi apoi să sustragă pozele ca fişiere JPEG, care evită orice filtru de texte.

Şi încă nu am discutat faptul că 70% din totalul atacurilor vin din interiorul reţelei protejate de zidul de protecţie, de exemplu, de la angajaţii nemulţumiţi (Schneier, 2000).

În plus, există o întreagă altă clasă de atacuri cărora zidurile de protecţie nu le pot face faţă. Ide- ea de bază a unui zid de protecţie este de a împiedica intruşii să pătrundă în reţeaua protejată şi de a împiedica datele secrete să iasă din acea reţea. Din păcate, există oameni care nu au altceva mai bun de făcut, decât să încerce să scoată din funcţiune anumite situri. Ei realizează acest lucru prin trimiterea unui număr foarte mare de pachete legitime către o ţintă, până când aceasta va fi scoasă din funcţiune datorită încărcării mari. De exemplu, pentru a distruge un sit de web, un intrus poate trimite un pachet TCP SYN pentru a stabili o conexiune. Ca urmare, situl va aloca, într-o tabelă, o intrare pentru acea conexiune şi va trimite ca răspuns un pachet SYN +ACK. Dacă intrusul nu răspunde intrarea din tabelă va fi ocupată pentru un interval de timp de până la câteva secunde, până când se va produce un timeout. Dacă intrusul va trimite mii de cereri de conexiune, toate intrările din tabelă vor fi ocupate astfel încât nu vor mai putea fi stabilite noi conexiuni legitime. Atacurile în care scopul intrusului este de a opri funcţionarea ţintei, în locul sustragerii de informaţii, sunt denumite atacuri **DoS (eng.: Denial of Service, rom.: Refuzul Serviciilor)**. În mod obişnuit, pachetele de cereri au o adresă sursă falsă pentru ca intrusul să nu poată fi detectat uşor.

O variantă şi mai pesimistă este aceea în care intrusul a pătruns deja în sute de calculatoare aflate în alte zone ale lumii, pe care apoi le comandă să atace aceeaşi ţintă în acelaşi timp. Nu numai că această abordare creşte forţa de atac a intrusului, dar ea reduce şansele ca ea să fie detectată, deoarece pachetele provin de la un mare număr de maşini aparţinând unor utilizatori ce nu sunt suspecţi. Un astfel de atac este denumit atac **DDoS (eng.: Distributed Denial of Service, rom.: Refuzul serviciilor realizat în mod distribuit)**. Împotriva acestui atac este greu de găsit o apărare. Chiar dacă maşina atacată poate recunoaşte rapid o cerere falsă, trece un anumit timp pentru procesarea şi ignorarea acestei cereri, iar dacă sosesc destul de multe cereri pe secundă procesorul va fi ocupat tot timpul cu tratarea acestora.

8.6.3 Reţele private virtuale

Multe companii au birouri răspândite în mai multe oraşe şi uneori în mai multe ţări. În trecut, înaintea apariţiei reţelelor de date publice, era obişnuită închirierea de către aceste companii a unor linii telefonice, aparţinând companiilor de

telefonie, între unele sau între toate perechile de locaţii ale birourilor. Unele companii încă mai fac acest lucru. O reţea alcătuită din calculatoarele unei companii şi liniile telefonice închiriate este denumită **reţea privată**. Un exemplu de reţea privată conectând trei birouri este arătată în fig. 8-30(a).

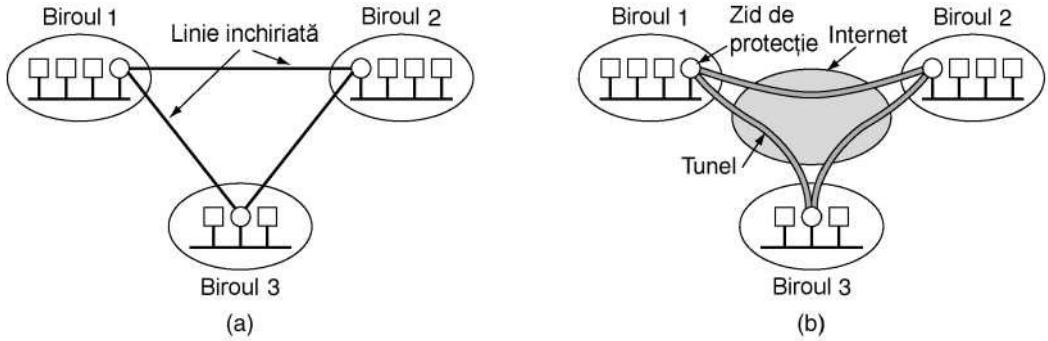


Fig. 8-30. (a) O reţea privată folosind linii închiriate (b) O reţea privată virtuală

Reţelele private funcţionează bine şi sunt foarte sigure. Dacă singurele linii disponibile sunt linii închiriate, atunci nu există trafic care să se scurgă în afara companiei, iar intruşii trebuie să se conecteze fizic la liniile respective pentru a pătrunde în reţea, ceea ce nu este un lucru uşor de realizat. Problema care apare cu reţelele private este acea că închirierea unei singure linii T1 costă mii de dolari pe lună, iar liniile T3 sunt de câteva ori mai scumpe. Când, mai târziu, au apărut reţelele de date publice şi Internetul, multe companii au vrut să-şi transmită traficul de date (şi posibil cel de voce) prin reţelele publice, dar fără a renunţa la securitatea unei reţele private.

Această necesitate a condus în curând la inventarea reţelelor private virtuale – **VPN (eng.: Virtual Private Networks, rom.: reţele private virtuale),** care sunt reţele construite deasupra unor reţele publice, dar care beneficiază de proprietăţile unei reţele private. Aceste reţele sunt denumite virtuale pentru că ele constituie o iluzie, aşa cum circuitele virtuale nu sunt circuite reale şi cum memoria virtuală nu este o memorie reală.

Deşi VPN-urile pot fi implementate peste ATM (sau frame relay), o tendinţă populară aflată în creştere este de a construi VPN-uri direct peste Internet. O modalitate de proiectare obişnuită este de a echipa fiecare birou cu un zid de protecţie şi de a crea tuneluri prin Internet între toate perechile de birouri, după cum este ilustrat în fig. 8-30(b). Dacă pentru tunelare este folosit IPsec, atunci este posibilă agregarea întregului trafic dintre oricare două perechi de birouri într-un singur SA autentificat şi criptat, oferindu-se astfel controlul integrităţii, confidenţialitate şi chiar o imunitate considerabilă asigurată analizei de trafic.

Când un sistem este pornit, fiecare pereche de ziduri de protecţie trebuie să negocieze parametrii SA-ului ei, incluzând serviciile, modurile, algoritmii şi cheile. Multe ziduri de protecţie au înglobate capabilităţi de VPN, deşi chiar şi unele rutere obişnuite beneficiază de acestea. Dar, deoarece zidurile de protecţie au prioritate în problema securităţii, este natural a avea tuneluri care încep şi se termină în ziduri

de protecţie, furnizând o separare clară între companie şi Internet. Astfel, zidurile de protecţie, VPN-urile şi IPsec folosit cu ESP în mod tunel formează o combinaţie naturală şi larg folosită în practică.

Odată ce a fost stabilit un SA, transferul datelor poate începe. Pentru un ruter din Internet, un pachet care traversează un tunel VPN este un pachet obişnuit. Singurul lucru neobişnuit la acesta este prezenţa antetului IPsec după antetul IP, dar deoarece aceste extra-antete nu au efect în procesul de rutare, ruterele nu le iau în considerare.

Un avantaj cheie al organizării VPN în acest fel este transparenţa completă pentru programele utilizatorilor. Zidurile de protecţie iniţializează şi gestionează SA-urile. Singura persoană care are cunoştinţă de acest proces este administratorul de sistem care configurează şi gestionează zidul de protecţie. Pentru oricine altcineva lucrurile arată ca şi cum ar fi o reţea privată bazată pe linie închiriată. Pentru mai multe despre VPN consultaţi (Brown, 1999 şi Izzo, 2000).

8.6.4 Securitatea în comunicaţiile fără fir

Este surprinzător de uşor de proiectat un sistem care din punct de vedere logic, este complet securizat folosind VPN şi ziduri de protecţie, dar prin care, în practică, informaţia se scurge ca prin sită. Această situaţie poate fi întâlnită dacă o parte dintre maşini sunt fără fir şi folosesc comunicaţia radio, care trece prin zidul de protecţie în ambele sensuri. Aria de acoperire a reţelelor 802.11 este foarte adesea de câteva sute de metri, astfel că oricine doreşte să spioneze o companie, poate să vină dimineaţa în parcarea pentru angajaţi, să lase un calculator portabil care este configurat pentru o reţea 802.11 să înregistreze tot ce aude şi să dispară pentru restul zilei. Până după-amiaza târziu, discul dur va fi plin de lucruri valoroase. Teoretic, această scurgere nu ar trebui să aibă loc. Teoretic, nici lumea nu ar trebui să jefuiască bănci.

O mare parte a problemei de securitate poate fi urmărită până la producătorii de staţii de bază care comunică fără fir (puncte de acces), care încearcă să-şi facă produsele prietenoase pentru utilizatori. De obicei, dacă utilizatorul scoate dispozitivul din cutie şi îl introduce în priză, acesta începe să funcţioneze imediat – aproape fără nici un fel de securitate, împrăştiind secrete către oricine se află în aria de acoperire radio. Dacă apoi este şi legat la o reţea Ethernet, tot traficul Ethernet va apărea dintr-o dată şi în locul de parcare. Reţelele fără fir sunt visul oricărui agent ascuns: date gratuite, fără să trebuiască să lucrezi pentru ele. Prin urmare, nici nu mai trebuie spus că securitatea este şi mai importantă pentru reţelele fără fir decât pentru cele cu fir. În această secţiune ne vom uita la câteva moduri în care reţelele fără fir tratează securitatea. Câteva informaţii suplimentare pot fi găsite în (Nicholas şi Lekkas, 2002).

Securitatea 802.11

Standardul 802.11 recomandă un protocol de securitate la nivelul legătură de date care se numeşte **WEP** (eng.: **Wired Equivalent Privacy** rom.: **Confidenţialitate Echivalentă cu cea Cablată**), care este proiectat pentru a face securitatea unui LAN cu comunicaţie fără fir la fel de bună ca cea a unui LAN cablat. Cum LAN-urile cu cabluri nu au de obicei nici un fel de securitate, acest obiectiv este uşor de

îndeplinit şi WEP îl îndeplineşte, după cum vom vedea.

Când securitatea 802.11 este activată, fiecare staţie are o cheie secretă comună cu staţia de bază. Standardul nu specifică cum sunt distribuite cheile. Pot fi încărcate înainte de către producător. Pot fi interschimbate înainte prin reţeaua cu fire. În fine, fie staţia de bază, fie maşina utilizatorului poate alege o cheie aleatoare şi o poate trimite prin aer criptată cu cheia publică a celeilalte staţii. Odată stabilită, cheia rămâne neschimbată în general pentru luni sau ani.

Criptarea WEP foloseşte un cifru flux bazat pe algoritmul RC4. RC4 a fost proiectat de Ronald Rivest şi a fost ţinut secret până în 1994 când s-au scurs nişte informaţii şi a fost publicat pe Internet. După cum am mai subliniat înainte, este aproape imposibil ca algoritmii să fie păstraţi secreţi, chiar dacă scopul este protejarea proprietăţii intelectuale (cum a fost în acest caz), mai degrabă decât securitatea prin obscuritate (care nu a fost scopul cu RC4). În WEP, RC4 generează un şir-cheie (eng.: keystream) care este combinat prin XOR cu textul clar pentru a forma textul cifrat.

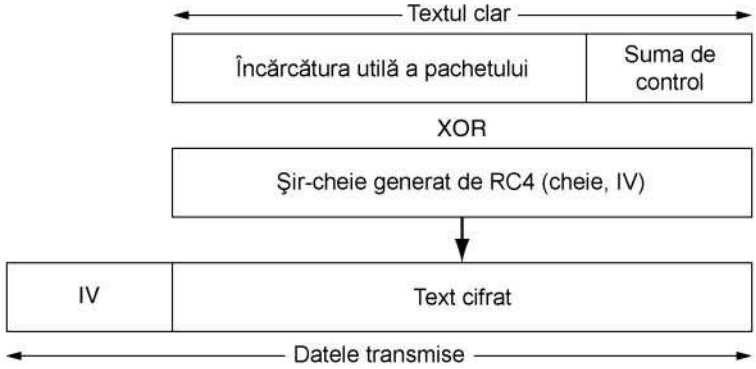


Fig. 8-31. Criptarea pachetelor folosind WEP

Încărcătura utilă a fiecărui pachet este criptată folosind metoda din fig. 8-31. Mai întâi, încărcătura este verificată folosind polinomul CRC-32, iar suma de control este adăugată încărcăturii pentru a forma textul clar pentru algoritmul de criptare. Acest text clar este combinat XOR cu o bucată din şirul cheie de aceeaşi lungime. Rezultatul este textul cifrat. IV-ul utilizat pentru pornirea RC4 este trimis împreună cu textul cifrat. Când receptorul primeşte pachetul, extrage din acesta încărcătura utilă criptată, generează şirul cheie din cheia secretă şi din IV pe care tocmai l-a primit şi apoi efectuează operaţia XOR între şirul cheie şi încărcătura utilă a pachetului pentru a obţine textul în clar. Apoi poate verifica suma de control pentru a se asigura că pachetul a ajuns intact.

Deşi această abordare arată bine la prima vedere, a fost deja publicată o metodă de a o sparge (Borisov et. al, 2001). Mai jos vom rezuma rezultatele ei. În primul rând, surprinzător de multe instalaţii folosesc aceeaşi cheie comună pentru toţi utilizatorii, caz în care fiecare utilizator poate citi traficul tuturor celorlalţi utilizatori. Acest lucru este desigur echivalent cu Ethernet-ul, dar nu este foarte sigur.

WEP poate fi atacat chiar dacă fiecare utilizator are o cheie distinctă,. Din

moment ce cheile sunt în general stabile pentru perioade lungi de timp, standardul WEP recomandă (dar nu impune) ca IV-ul să fie schimbat la fiecare pachet pentru a evita atacul de tip reutilizare asupra şirului cheie, atac discutat în secţiunea 8.2.3. Din nefericire, multe plăci 802.11 pentru calculatoarele portabile resetează IV la 0 când placa este introdusă în calculator şi apoi îl incrementează cu 1 la fiecare pachet trimis. Cum utilizatorii scot şi apoi reinserează frecvent aceste plăci, pachetele cu valori mici pentru IV sunt destul de obişnuite. Dacă Trudy poate să colecteze mai multe pachete trimise de acelaşi utilizator, care au aceeaşi valoare pentru IV (care este trimis în clar cu fiecare pachet), atunci ea poate să calculeze XOR între două valori de textul clar şi poate astfel să spargă cifrul.

Dar, chiar dacă placa 802.11 alege o valoare aleatoare pentru fiecare pachet, IV are doar 24 de biţi, astfel încât după ce au fost trimise 2^{24} pachete, va trebui ca valorile să fie refolosite. Mai rău, folosind valori aleatoare pentru IV, numărul probabil de pachete care trebuie trimise înainte ca acelaşi număr să fie folosit de două ori este în jur de 5000, datorită atacului de tip „ziua de naştere” descris în secţiunea 8.4.4. Ca urmare, dacă Trudy ascultă pentru câteva minute, ea este aproape sigură că va captura două pachete cu acelaşi IV şi aceeaşi cheie. Efectuând operaţia XOR între cele două texte cifrate, ea este capabilă să obţină combinaţia XOR dintre textele în clar. Această secvenţă de biţi poate fi atacată în diferite moduri pentru a descoperi textele în clar. Cu mai multă muncă, şirul cheie pentru acel IV poate fi obţinut de asemenea. Trudy poate continua să lucreze în acest mod pentru a realiza un dicţionar de şiruri cheie pentru diferite IV. Odată spart un IV, toate pachetele trimise cu acesta în viitor (dar şi în trecut) pot fi complet decriptate.

Mai mult, deoarece IV-rile sunt folosite aleator, odată ce Trudy a determinat o pereche (IV, şir- cheie) validă, ea o poate folosi pentru a genera toate pachetele pe care le doreşte şi astfel, să intervină activ în comunicaţie. Teoretic, un receptor ar putea observa că dintr-o dată un număr mare de pachete au toate acelaşi IV, dar (1) WEP permite acest lucru şi (2) oricum nimeni nu verifică acest lucru.

În sfârşit, CRC-ul nu valorează prea mult, deoarece este posibil ca Trudy să modifice încărcătura utilă şi apoi să facă schimbările corespunzătoare în CRC, fără a trebui măcar să elimine criptarea. Pe scurt, spargerea securităţii 802.11 este destul de evidentă, şi nici măcar nu am enumerat toate atacurile pe care le-au găsit Borisov et. al.

În august 2001, la o lună după ce a fost prezentată lucrarea lui Borisov et. al, a fost publicat un alt atac devastator asupra WEP (Fluhrer et. al, 2001). Acesta a găsit slăbiciuni criptografice chiar în RC4. Fluhrer et. al au descoperit că multe dintre chei au proprietatea că este posibilă deducerea câtorva dintre biţii cheii din şirul-cheie. Dacă acest atac este pus în aplicare repetat, este posibilă deducerea întregii chei cu un efort modest. Fiind înclinaţi mai mult către teorie, Fluhrer et. al nu au încercat efectiv să spargă vreun LAN 802.11.

În contrast, când un student la cursurile de vară şi doi cercetători de la AT&T Labs au aflat despre atacul lui Fluhrer et. al, s-au decis să îl încerce într-un caz real (Stubblefield et. al, 2002). Într-o săptămână ei au spart prima lor cheie de 128 de biţi dintr-un LAN 802.11 de producţie şi cea mai mare parte a săptămânii a fost de fapt dedicată căutării celei mai ieftine plăci de reţea 802.11, obţinerii permisiunii de a o cumpăra, instalării şi testării ei. Programarea a durat efectiv doar două ore.

Când şi-au anunţat rezultatele, CNN a emis o ştire intitulată „Un spărgător de rutină sparge criptarea comunicaţiilor fără fir”, în care nişte guru din industrie au încercat să minimalizeze rezultatele lor, spunând că ceea ce au făcut ei a fost trivial, date fiind rezultatele lui Fluhrer et. al Chiar dacă această observaţie este tehnic adevărată, rămâne faptul că eforturile combinate ale acestor două echipe au demonstrat o scăpare fatală în WEP şi 802.11.

În 7 Septembrie, 2001, IEEE a răspuns faptului că WEP era atunci complet spartă printr-o declaraţie scurtă cu şase puncte care pot fi rezumate grosier în felul următor:

1. Noi v-am spus că securitatea WEP nu era mai bună decât cea a Ethernet-ului.
2. O ameninţare mult mai mare este să uiţi complet să activezi securitatea.
3. Încercaţi să folosiţi altă securitate (de ex., securitatea nivelului transport)
4. Versiunea următoare, 802.11i, va avea o securitate mai bună.
5. Certificările ulterioare vor impune folosirea 802.11i.
6. Vom încerca să ne dăm seama ce se poate face până când apare 802.11i.

Am parcurs în detaliu toată această poveste pentru a demonstra faptul că o securitate solidă nu este uşor de obţinut, nici pentru experţi.

Securitatea Bluetooth

Bluetooth are o arie de acoperire mult mai mică decât 802.11 şi nu poate fi atacată din locul de parcare, dar securitatea este şi aici o problemă. De exemplu, imaginaţi-vă calculatorul lui Alice care are o tastatură fără fir Bluetooth. În absenţa securităţii, dacă Trudy este întâmplător în biroul de alături, ea ar putea citi tot ce scrie Alice, inclusiv toată corespondenţa electronică trimisă. De asemenea, ea ar putea captura tot ce trimite calculatorul lui Alice imprimantei Bluetooth care se află lângă ea (corespondenţa electronică primită şi rapoarte confidenţiale). Din fericire, Bluetooth are o schemă de securitate pentru a încerca să le învingă pe orice Trudy din lume. Vom rezuma în continuare principalele trăsături ale acesteia.

Bluetooth are trei moduri de securitate, de la nici o securitate la criptarea tuturor datelor şi controlul integrităţii. La fel ca şi 802.11, dacă securitatea este dezactivată (opţiunea care este selectată iniţial) atunci nu există securitate. Majoritatea utilizatorilor au securitatea dezactivată până când are loc o spargere; apoi o activează. În lumea agriculturii, această abordare este cunoscută drept încuie- rea uşii hambarului după ce a fugit calul.

Bluetooth oferă securitate pe mai multe niveluri. La nivelul fizic, salturile în frecvenţă oferă un pic de securitate, dar deoarece oricare dispozitiv Bluetooth care se mişcă într-un piconet trebuie informat de secvenţa de salturi în frecvenţă, această secvenţă evident că nu este un secret. Adevărata securitate începe când un sclav (eng.: slave) nou-venit cere un canal cu stăpânul (eng.: master). Este de presupus că cele două dispozitive împart o cheie secretă stabilită anterior. În unele cazuri, cheia este fixată în ambele dispozitive de către producător (de ex., pentru un set de căşti cu microfon şi un telefon mobil vândute împreună). În alte cazuri, un dispozitiv (de ex., setul de căşti) are o cheie fixă, iar utilizatorul trebuie să introducă acea cheie în celălalt dispozitiv (de ex., telefonul mobil) ca un număr zecimal. Aceste chei comune se numesc **chei de trecere** (eng.: **passkeys**).

Pentru a stabili un canal, sclavul şi stăpânul verifică fiecare dacă celălalt

cunoaşte cheia de trecere. Dacă da, ei negociază dacă acel canal va fi criptat, cu controlul integrităţii, sau amândouă. Apoi ei aleg o cheie de sesiune aleatoare de 128 de biţi, dintre care câţiva ar putea fi publici. Ideea acestei slăbiri a cheii este alinierea la restricţiile guvernamentale din diferite ţări proiectate, pentru a preveni exportul, sau utilizarea cheilor mai lungi decât acelea pe care guvernul le poate sparge.

Criptarea foloseşte un cifru-flux denumit $\mathbf{E}_0$; controlul integrităţii foloseşte **SAFER+**. Amândouă sunt cifruri bloc tradiţionale cu chei simetrice. SAFER+ a fost înscris în concursul AES, dar a fost eliminat în prima rundă pentru că era mai lent decât ceilalţi candidaţi. Bluetooth a fost finalizat înainte ca cifrul AES să fie ales; altfel, foarte probabil, ar fi folosit Rijndael.

Criptarea actuală folosind cifrul-flux este prezentată în fig. 8-14, cu textul în clar combinat prin XOR cu şirul-cheie pentru a genera textul cifrat. Din nefericire, E_0 însuşi (ca şi RC4) ar putea avea slăbiciuni fatale (Jakobsson şi Wetzel, 2001). Deşi nu a fost spart până la momentul scrierii acestei cărţi, similitudinile lui cu cifrul A5/1, al cărui eşec spectaculos compromite tot traficul telefonic GSM, sunt o cauză de îngrijorare (Biryukov et. al, 2000). Uneori este uimitor pentru unii (incluzând autorul) că în perenul joc de-a şoarecele şi pisica între criptografi şi criptanalişti, criptanalişti sunt atât de des în partea învingătoare.

O altă problemă de securitate este că Bluetooth autentifică doar dispozitivele, nu utilizatorii, astfel că furtul unui dispozitiv Bluetooth poate permite hoţului accesul la contul financiar şi la alte conturi ale utilizatorului. Oricum, Bluetooth implementează de asemenea securitatea pentru nivelele înalte, astfel că în cazul unei spargeri a securităţii de la nivelul legătură, securitatea se menţine, în special pentru aplicaţiile care necesită introducerea manuală a unui cod de PIN de la un fel de tastatură pentru a completa tranzacţia.

Securitatea WAP 2.0

În mare parte, Forumul WAP a învăţat ceva din greşeala de a avea o stivă de protocoale nestandard în WAP 1.0. WAP 2.0 foloseşte generos protocoalele standard la toate nivelurile. Securitatea nu este o excepţie. Deoarece este bazat pe IP, el suportă folosirea integrală a IPsec la nivelul reţea. La nivelul transport, conexiunile TCP pot fi protejate de TLS, un standard IETF pe care îl vom studia mai târziu în acest capitol. Şi mai sus, foloseşte autentificarea HTTP a clientului, aşa cum e definită în RFC 2617. Bibliotecile criptografice de la nivelul aplicaţie oferă controlul integrităţii şi nerepudierea. Una peste alta, din moment ce WAP 2.0 este bazat pe standarde bine cunoscute, există o şansă ca serviciile sale de securitate, în particular, confidenţialitatea, controlul integrităţii şi nerepudierea să valoreze mai mult decât securitatea 802.11 şi Bluetooth.

8.7 PROTOCOALE DE AUTENTIFICARE

Autentificarea (eng.: *authentication)* este tehnica prin care un proces verifică dacă partenerul său de comunicaţie este cel presupus a fi şi nu un impostor.

Verificarea identităţii unui proces de la distanţă, în cazul unui intrus activ şi răuvoitor, este surprinzător de dificilă şi necesită protocoale complexe bazate pe criptografie. În această secţiune, vom studia câteva dintre numeroasele protocoale de autentificare folosite în reţelele nesigure de calculatoare.

Ca fapt divers, anumiţi oameni confundă autorizarea cu autentificarea. Autentificarea se ocupă cu problema de a şti dacă într-adevăr comunici cu un anumit proces. Autorizarea se ocupă cu ceea ce îi este permis unui proces să facă. De exemplu, un proces client contactează un server de fişiere şi spune: „Eu sunt procesul lui Scott şi vreau să şterg fişierul *cookbook.old".* Din punctul de vedere al serverului de fişiere, trebuie găsite răspunsurile la două întrebări:

1. Chiar este procesul lui Scott? (autentificare)
2. Are Scott permisiunea de a şterge *cookbook.old?* (autorizare)

Doar după ce s-a răspuns afirmativ, fără ambiguităţi, la ambele întrebări poate avea loc acţiunea cerută. De fapt prima dintre cele două este cu adevărat întrebarea cheie. Odată ce serverul de fişiere
ştie cu cine vorbeşte, verificarea autorizării este doar o problemă de căutare în intrările tabelelor şi bazelor de date locale. Din acest motiv, în această secţiune ne vom concentra asupra autentificării.

Modelul general pe care îl folosesc toate protocoalele de autentificare este următorul. Alice începe prin a trimite un mesaj fie lui Bob, fie unui centru autorizat de distribuire a cheilor **KDC** (eng.: **Key Distribution Center**), care este presupus a fi credibil. Urmează alte câteva schimburi de mesaje în diferite direcţii. În timp ce aceste mesaje sunt transmise, Trudy, le poate intercepta, modifica sau retrimite în scopul de a-i înşela pe Alice şi Bob sau doar pentru a încurca lucrurile. Cu toate acestea, la încheierea protocolului de comunicare, Alice este sigură că a vorbit cu Bob, iar acesta este sigur că a vorbit cu Alice. Mai mult decât atât, în cele mai multe protocoale, cei doi vor fi stabilit şi o **cheie secretă de sesiune**, pentru folosirea în conversaţiile viitoare. În practică, din motive de performanţă, tot traficul de date este criptat folosind criptografia cu cheie secretă (de obicei AES sau DES triplu), în timp ce criptografia cu cheie publică este larg folosită în protocoale de autentificare şi pentru stabilirea unei chei de sesiune.

Motivul pentru utilizarea unei chei de sesiune noi, alese aleator, pentru fiecare conexiune, este minimizarea mărimii traficului care este transmis cu cheile secrete sau publice ale utilizatorilor, pentru a reduce cantitatea de text cifrat pe care intrusul o poate obţine şi pentru a reduce pagubele care se pot produce dacă un proces eşuează, iar vidajul său de memorie (eng.: core dump) cade în mâini rele. Din fericire, singura cheie prezentă va fi cheia de sesiune. Toate cheile permanente trebuie să fie anulate cu grijă după stabilirea sesiunii.

8.7.1 Autentificare bazată pe cheie secretă partajată

Pentru primul nostru protocol de autentificare, vom presupune că Alice şi Bob partajează deja o cheie secretă, K_{AB} Această cheie partajată poate să fi fost stabilită anterior fie prin telefon, fie direct, dar în nici un caz nu prin intermediul unei reţele (nesigure).

Protocolul este bazat pe un principiu ce poate fi întâlnit în multe protocoale de autentificare: o parte trimite un număr aleatoriu celeilalte, care îl transformă apoi într-un anumit mod şi returnează rezultatul. Astfel de protocoale se numesc protocoale **provocare-răspuns** (eng.: *challenge-response).* În acest protocol şi în cele ce urmează va fi folosită următoarea notaţie:

A, *B* reprezintă identităţile lui Alice şi Bob.
R_i reprezintă provocările, unde indicele identifică pe cel ce trimite provocarea.
K_i sunt cheile, unde *i* indică proprietarul.
K_S este cheia de sesiune.

Secvenţa de mesaje pentru primul nostru protocol de autentificare cu cheie partajată este ilustrată în fig. 8-32. În mesajul 1 Alice îi trimite lui Bob identitatea sa, A, într-un mod pe care Bob îl înţelege. Bineînţeles că Bob nu are posibilitatea de a şti dacă acest mesaj vine de la Alice sau de la Trudy, astfel încât el alege o provocare, un număr aleator mare, R_B, şi îl trimite înapoi spre „Alice” sub forma mesajului 2, ca text clar. Alice criptează atunci mesajul cu cheia pe care ea o partajează cu Bob şi trimite textul cifrat $K_{AB}(R_B)$ înapoi, ca mesajul 3. Când Bob vede acest mesaj el ştie imediat că vine de la Alice, deoarece Trudy nu cunoaşte K_{AB} şi astfel nu putea să genereze mesajul respectiv. Mai mult decât atât, deoarece R_B a fost ales aleator dintr-un spaţiu mare (să spunem, de exemplu, numere aleatoare pe 128 de biţi), este destul de puţin probabil ca Trudy să fi văzut R_B şi răspunsul asociat lui într-o sesiune anterioară. Este la fel de puţin probabil ca ea să fi putut ghici răspunsul corect la orice provocare.

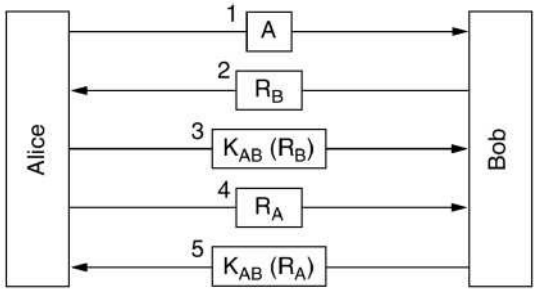


Fig. 8-32. Autentificare în doi paşi folosind un protocol de tipul

În acest moment, Bob este sigur că vorbeşte cu Alice, dar Alice nu este sigură de nimic. Din punctul ei de vedere, s-ar putea ca Trudy să fi interceptat mesajul 1 şi să fi trimis înapoi replica *RB.* Poate că Bob a murit azi-noapte. Pentru a descoperi cu cine vorbeşte, Alice alege un număr aleator R_A şi îl trimite lui Bob ca text clar, în mesajul 4. Când Bob răspunde cu $K_{AB}(R_A)$, Alice ştie că vorbeşte cu Bob. Dacă ei vor să stabilească acum o cheie de sesiune, Alice poate alege una, K_s, şi o poate trimite lui Bob criptată cu K_{AB}.

Protocolul din fig. 8-32 conţine cinci mesaje. Să vedem dacă putem să fim isteţi şi să eliminăm o parte din ele. O variantă este ilustrată în fig. 8-33. Aici Alice iniţiază protocolul provocare-răspuns, în loc de a-l aştepta pe Bob să o facă. Similar, în timp ce răspunde la provocarea lui Alice, Bob o trimite pe a sa. Întregul protocol poate fi redus la trei mesaje în loc de cinci.

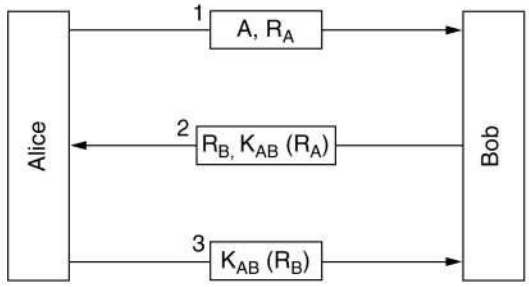


Fig. 8-33. Un protocol mai scurt de autentificare în doi paşi.

Este acest nou protocol o îmbunătăţire a celui original? Într-un anumit sens da: este mai scurt. Din nefericire, este şi greşit. În anumite situaţii, Trudy poate învinge acest protocol folosind ceea ce se numeşte **atacul prin reflexie**. În particular, Trudy îl poate sparge dacă este posibil să deschidă sesiuni multiple cu Bob simultan. Această situaţie ar fi adevărată, de exemplu, dacă Bob este o bancă şi este pregătit să accepte simultan mai multe conexiuni cu maşini de efectuat plăţi.

Atacul prin reflexie al lui Trudy este prezentat în fig. 8-34. Pentru început, Trudy pretinde că este Alice şi trimite R_T. Bob răspunde, ca de obicei, cu propria sa provocare, R_B. Acum Trudy este în impas. Ce poate să facă? Nu cunoaşte $K_{AB}(R_B)$.

Ea poate deschide o a doua sesiune cu mesajul 3, furnizând R_B luat din mesajul 2 ca provocare a ei. Bob îl criptează calm şi trimite înapoi K_{AB} (R_B) în mesajul 4. Am colorat mesajele din sesiunea a doua pentru a le evidenţia. Acum Trudy are informaţia care îi lipsea, aşa că poate să finalizeze prima sesiune şi să o abandoneze pe cea de-a doua. Bob este acum convins că Trudy este Alice, aşa că atunci când ea

întreabă de balanţa contului său bancar, el i-o dă fără nici o problemă. Apoi când ea îi cere să transfere toţi banii la o bancă secretă din Elveţia, el face asta fără nici un moment de ezitare.

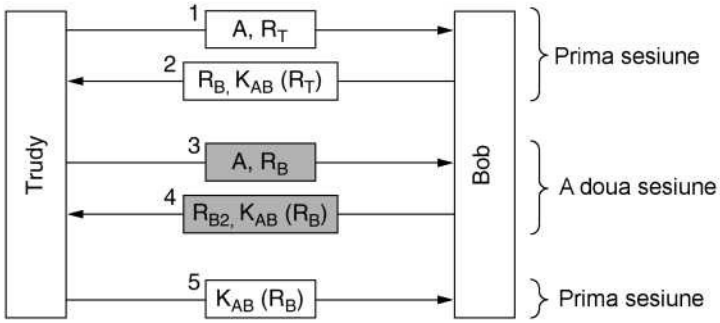


Fig. 8-34. Atacul prin reflexie.

Morala aceste povestiri este:

A proiecta un protocol corect de autentificare este mai greu decât pare.

Următoarele patru reguli generale sunt adeseori utile:

1. Iniţiatorul să dovedească cine este înaintea celui care răspunde. În acest caz, Bob transmite informaţii importante înainte ca Trudy să-i fi dat vreo dovadă că este cine pretinde a fi.
2. Iniţiatorul şi cel ce răspunde să folosească chei diferite pentru dovadă, chiar dacă aceasta presupune existenţa a două chei partajate K_B şi K'_{AB}.
3. Iniţiatorul şi cel ce răspunde să-şi extragă provocările din mulţimi diferite. De exemplu, iniţiatorul trebuie să folosească numere pare, iar cel ce răspunde să folosească numere impare.
4. Protocolul să fie rezistent la atacuri ce implică o a doua sesiune paralelă, în care informaţia obţinută într-o sesiune este folosită într-o alta.

Dacă fie şi una singură din aceste patru reguli este nerespectată, deseori protocolul poate fi spart. În acest caz, patru reguli au fost încălcate cu consecinţe dezastroase.

Acum să ne întoarcem şi să ne uităm mai bine la protocolul din fig. 8-32. Sigur acel protocol nu poate fi ţinta unui atac prin reflexie? Depinde. Este destul de subtil. Trudy a reuşit să învingă protocolul nostru folosind un atac prin reflexie pentru că a fost posibil să deschidă o a doua sesiune cu Bob şi să îl păcălească, făcându-l să răspundă la propriile întrebări. Ce s-ar întâmpla dacă Alice ar fi un calculator de uz general care ar accepta şi el sesiuni multiple, şi nu o persoană în faţa unui calculator? Să vedem ce poate face Trudy.

Pentru a vedea cum funcţionează atacul lui Trudy, a se vedea fig. 8-35. Alice începe prin a-şi anunţa identitatea în mesajul 1. Trudy interceptează acest mesaj şi îşi începe propria sesiune cu mesajul 2, pretinzând că este Bob. Am colorat din nou mesajele din sesiunea a doua. Alice răspunde la mesajul 2 spunând: Pretinzi că eşti Bob? Demonstrează. în mesajul 3. În acest moment Trudy e în impas pentru că nu poate demonstra că este Bob.

Ce face Trudy acum? Se întoarce la prima sesiune, în care e rândul ei să trimită

o provocare, şi trimite R_A pe care a primit-o în mesajul 3. Alice îi răspunde politicos în mesajul 5, furnizându-i lui Trudy informaţia de care are nevoie pentru a trimite mesajul 6 din sesiunea 2. În acest moment Trudy este practic liberă pentru că a răspuns cu succes la provocarea lui Alice în sesiunea 2. Acum poate să abandoneze sesiunea 1, să trimită un număr oarecare pentru restul sesiunii 2 şi va avea o sesiune autentificată cu Alice în sesiunea 2.

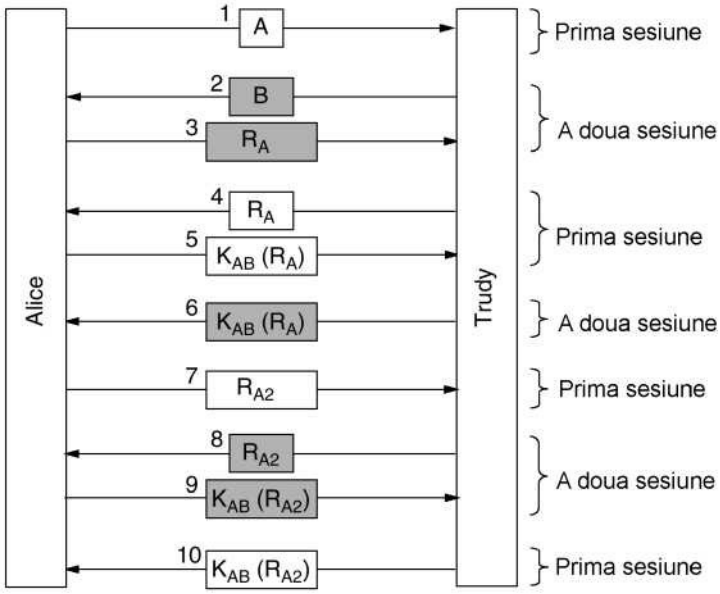


Fig. 8-35. Atac prin reflexie asupra protocolului din fig. 8-32.

Dar Trudy e periculoasă şi chiar vrea să pună sare pe rană. În loc să trimită un număr oarecare pentru a completa sesiunea 2, ea aşteaptă ca Alice să trimită mesajul 7, provocarea lui Alice pentru sesiunea 1. Bineînţeles, Trudy nu ştie cum să răspundă, aşa că foloseşte din nou atacul prin reflexie, trimiţând R_{A2} ca mesajul 8. Alice criptează R_{A2} în mesajul 9. Acum Trudy se întoarce la sesiunea 1 şi îi trimite lui Alice numărul dorit în mesajul 10, copiat din ceea ce trimisese Alice în mesajul 9. Acum Trudy are două sesiuni complet autentificate cu Alice.

Acest atac are un rezultat oarecum diferit de atacul protocolului cu trei mesaje reprezentat în fig. 8-34. De această dată, Trudy are două sesiuni autentificate cu Alice. În exemplul anterior, ea avea o singură sesiune autentificată cu Bob. Şi aici, dacă am fi aplicat toate regulile generale pentru protocoale de autentificare discutate mai sus, acest atac ar fi putut fi oprit. O discuţie detaliată despre acest gen de atacuri şi despre cum să le împiedicăm se găseşte în (Bird et. al, 1993). Ei arată şi cum este posibilă construirea sistematică de protocoale pentru care să se poată demonstra că sunt corecte. Cel mai simplu protocol de acest fel este totuşi cam complicat, aşa că acum vom prezenta o altă clasă de protocoale care funcţionează de asemenea corect.

Noul protocol de autentificare este reprezentat în fig. 8-36 (Bird ş. a., 1993). Acesta foloseşte un HMAC de tipul pe care l-am văzut când am studiat IPsec. Alice începe prin a-i trimite lui Bob un număr ad-hoc, R_A, ca mesajul 1. Bob răspunde

alegând propriul număr ad-hoc, R_B, şi trimiţându-l înapoi împreună cu un HMAC. HMAC-ul este format prin construirea unei structuri de date alcătuită din numărul ad-hoc al lui Alice, numărul ad-hoc al lui Bob, identităţile lor şi cheia secretă partajată, K_{AB} Această structură de date este codificată prin dispersie în HMAC, de exemplu folosind SHA-1. Când Alice primeşte mesajul 2, ea are R_A (pe care a ales-o ea însăşi), R_B, care ajunge ca text clar, cele două identităţi, şi cheia secretă, K_{AB}, pe care a ştiut-o tot timpul, aşa că poate să calculeze singură HMAC-ul. Dacă acesta coincide cu cel din mesaj, ea ştie că vorbeşte cu Bob deoarece Trudy nu cunoaşte K_{AB}, deci nu-şi poate da seama ce HMAC să trimită. Alice îi răspunde lui Bob cu un HMAC ce conţine doar cele două numere ad-hoc.

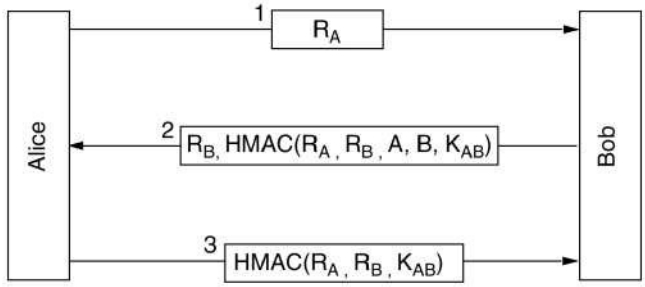


Fig. 8-36. Autentificarea folosind HMAC.

Poate Trudy să spargă în vreun fel acest protocol? Nu, pentru că nu poate forţa nici una din părţi să cripteze sau să rezume o valoare aleasă de ea, cum se întâmpla în fig. 8-34 şi în fig. 8-35. Ambele HMAC-uri includ valori alese de partea care le trimite, ceea ce Trudy nu poate controla.

Utilizarea HMAC-urilor nu este singurul mod de a folosi această idee. O schemă alternativă deseori utilizată în locul calculării HMAC pentru o serie de elemente este criptarea acestor elemente secvenţial, folosind înlănţuirea blocurilor cifrate.

8.7.2 Stabilirea unei chei secrete: schimbul de chei Diffie-Hellman

Până acum am presupus că Bob şi Alice partajează o cheie secretă. Să presupunem că nu este aşa (pentru că deocamdată nu există o PKI universal acceptată pentru semnarea şi distribuirea certificatelor). Cum pot ei să stabilească una? O modalitate ar fi ca Alice să-l sune pe Bob şi să-i dea cheia ei prin telefon, dar el probabil va începe întrebând: Cum ştiu eu că eşti Alice şi nu Trudy? Ei ar putea încerca să aranjeze o întâlnire la care fiecare din ei să aducă un paşaport, un permis de conducere şi trei cărţi de credit semnificative, dar fiind oameni ocupaţi, e posibil să nu poată găsi, vreme de luni de zile, o dată acceptabilă pentru amândoi. Din fericire, oricât de incredibil ar părea, există un mod ca cei ce sunt total străini să stabilească, chiar la lumina zilei, o cheie secretă partajată, cu Trudy înregistrând grijulie fiecare mesaj.

Protocolul care permite străinilor să stabilească o cheie secretă partajată se numeşte **interschimbul de chei Diffie-Hellman** (Diffie şi Hellman, 1976) şi funcţionează după cum urmează. Alice şi Bob trebuie să se pună de acord asupra a două numere mari, n şi g, unde n este prim, (n-1)/2 este de asemenea prim şi g îndeplineşte anumite condiţii. Aceste numere pot fi publice, astfel că oricare din ei poate pur şi simplu să aleagă n şi g şi să o spună celuilalt, în mod deschis. Acum Alice alege un număr mare (să spunem pe 512 biţi), x, şi îl păstrează secret. Similar, Bob alege un număr mare, secret, y.

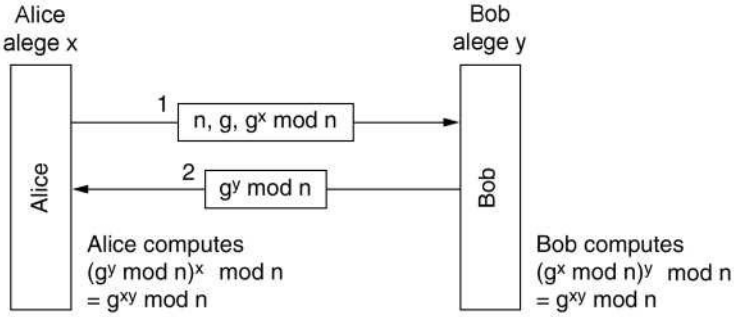


Fig. 8-37. Schimbul de cheie Diffie-Hellman.

Alice iniţiază protocolul de schimb al cheii trimiţându-i lui Bob un mesaj ce conţine *(n, g, g* mod *n),* după cum se arată în fig. 8-37. Bob răspunde trimiţându-i lui Alice un mesaj ce conţine *g* mod *n*. Acum Alice are numărul pe care i l-a trimis Bob şi îl ridică la puterea *x* modulo n pentru a obţine *(g* mod *n)*^{x} mod *n*. Bob efectuează o operaţie similară pentru a obţine (*g*^{x} mod *n)* mod *n*. Conform legilor aritmeticii modulare, ambele calcule duc la *g*^{xy} mod *n*. Iată cum Alice şi Bob partajează acum o cheie secretă *g*^{x} mod *n*.

Este evident că Trudy vede ambele mesaje. Ea cunoaşte pe *g* şi pe *n* din mesajul 1. Dacă ea ar putea calcula *x* şi *y* ar putea să descopere cheia secretă. Necazul este că, dat fiind doar *g*^{x} mod *n*, ea nu poate afla pe *x*. Nu este cunoscut nici un algoritm practic pentru calculul logaritmilor discreţi modulo un număr prim foarte mare.

Pentru a face exemplul anterior mai concret, vom folosi (complet nerealist) valorile *n* = 47 şi *g* =
3. Alice alege *x* = 8 şi Bob alege *y* = 10. Ambele chei sunt păstrate secrete. Mesajul lui Alice către Bob este (47, 3, 28) deoarece 3^8 mod 47 este 28. Mesajul lui Bob către Alice este (17). Alice calculează 17^8 mod 47, care este 4. Bob calculează 28^{10} mod 47, care este 4. Alice şi Bob au determinat independent cheia secretă care este 4. Trudy are de rezolvat ecuaţia 3^x mod 47 = 28, ceea ce poate fi făcut prin căutare exhaustivă în cazul unor numere mici ca acestea, dar nu şi atunci când toate numerele sunt lungi de sute de biţi. Toţi algoritmii cunoscuţi la ora actuală iau prea mult timp, chiar şi atunci când sunt rulaţi folosind un supercalculator masiv paralel.

În ciuda eleganţei algoritmului Diffie-Hellman, există o problemă: când Bob ia tripletul (47, 3, 28), cum ştie el că este de la Alice şi nu de la Trudy? Nu există nici o modalitate pentru aceasta. Din nefericire Trudy poate exploata acest fapt pentru a-i înşela atât pe Alice cât şi pe Bob, după cum este ilustrat în fig. 8-38. Aici, când Alice şi Bob îl aleg pe *x*, respectiv pe *y*, Trudy alege propriul său număr aleator, *z*. Alice trimite mesajul 1, destinat lui Bob. Trudy îl interceptează şi trimite mesajul 2 lui Bob, folosind *g* şi *n* corecţi (care sunt disponibili public), dar cu al său *z* în loc de *x*. De asemenea ea trimite mesajul 3 lui Alice. Mai târziu Bob îi trimite lui Alice mesajul 4, pe care Trudy îl interceptează din nou şi îl păstrează.

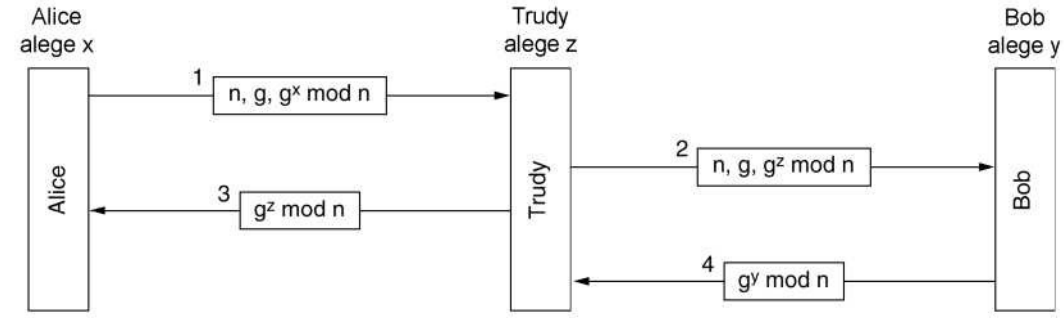


Fig. 8-38. Atacul de tip găleata brigăzii sau omul-din-mijloc.

Acum fiecare efectuează aritmetica modulară. Alice calculează cheia secretă ca fiind g^{xz} mod n şi la fel face şi Trudy (pentru mesajele lui Alice). Bob calculează g^{yz} mod n şi la fel face şi Trudy (pentru mesajele lui Bob). Alice crede că vorbeşte cu Bob, aşa că ea stabileşte o sesiune de cheie (cu Trudy). La fel face şi Bob. Fiecare mesaj pe care Alice îl trimite în sesiunea criptată este capturat de Trudy, memorat, modificat la dorinţă şi apoi (opţional) transmis lui Bob. Similar în cealaltă direcţie. Trudy vede orice şi poate modifica toate mesajele la dorinţă, în timp ce atât Alice cât şi Bob trăiesc cu iluzia că au un canal de comunicaţie sigur de la unul la celălalt. Acest atac este cunoscut sub numele de **atacul găleata brigăzii de pompieri** (eng.: *bucket brigade attack),* deoarece el seamănă vag cu un departament de pompieri de pe vremuri trecând din mână în mână găleţile de-a lungul drumului de la maşina de pompieri la foc. El se mai numeşte şi **atacul omul-din-mijloc** (eng.: *man-în-the-middle attack).*

8.7.3 Autentificarea folosind un Centru de Distribuţia Cheilor

Stabilirea unui secret partajat cu un străin a mers destul de bine, dar nu în întregime. Pe de altă parte, probabil că nici nu merită să fie făcut (atacul strugurilor acri). Pentru a vorbi cu n oameni în acest mod ar fi necesare n chei. Pentru persoanele foarte cunoscute, gestiunea cheilor ar deveni o adevărată pacoste, în special dacă fiecare cheie trebuie stocată separat pe câte o cartelă de plastic.

O abordare diferită o reprezintă introducerea unui centru autorizat de distribuţie a cheilor (KDC – Key Distribution Center). În acest model, fiecare utilizator are o singură cheie partajată cu KDC. Autentificarea şi gestiunea cheilor de sesiune merg acum prin intermediul KDC. Cel mai simplu protocol cunoscut pentru autentificarea KDC, implicând două părţi şi un centru autorizat, este ilustrat în fig. 8-39.

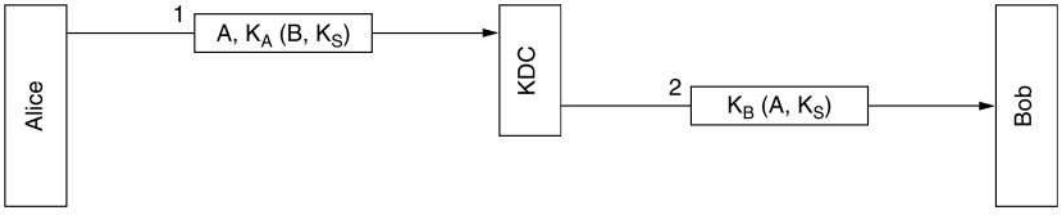


Fig. 8-39. O primă încercare de protocol de autentificare folosind un KDC .

Ideea din spatele protocolului este simplă: Alice alege o cheie de sesiune , K_S, şi

anunţă KDC că vrea să vorbească cu Bob folosind K_S. Acest mesaj este criptat cu cheia secretă pe care Alice o împarte (numai) cu KDC, K_A. KDC decriptează acest mesaj, extrage identitatea lui Bob şi cheia de sesiune. Apoi el construieşte un nou mesaj ce conţine identitatea lui Alice şi cheia de sesiune şi trimite acest mesaj lui Bob. Criptarea este făcută cu K_B, cheia secretă pe care Bob o împarte cu KDC. Când Bob decriptează mesajul, el află că Alice vrea să vorbească cu el şi cheia pe care aceasta vrea să o utilizeze.

Autentificarea are loc gratuit. KDC ştie că mesajul 1 trebuie să fi venit de la Alice, deoarece nimeni altcineva nu poate să-l cripteze cu cheia secretă a Alicei. Similar, Bob ştie sigur că mesajul 2 vine de la KDC, în care el are încredere, deoarece nimeni altcineva nu mai cunoaşte cheia lui secretă.

Din nefericire, acest protocol prezintă un defect grav. Trudy are nevoie de ceva bani, aşa că ea imaginează un serviciu pe care îl poate executa pentru Alice, face o ofertă atractivă şi obţine postul. După ce îşi face treaba, Trudy cere politicos lui Alice să-i plătească transferându-i banii prin bancă. Aşa că Alice stabileşte o cheie de sesiune cu bancherul ei, Bob. Apoi ea îi trimite lui Bob un mesaj prin care cere ca banii respectivi să fie transferaţi în contul lui Trudy.

Între timp, Trudy se întoarce la vechile ei obiceiuri, furturile prin reţea. Ea copiază atât mesajul 2 din fig. 8-39, cât şi cererea de transferare a banilor care îl urmează. Mai târziu ea le trimite lui Bob. Bob le ia şi gândeşte: „Alice probabil că a angajat-o din nou pe Trudy. Cu siguranţă că ea lucrează bine.” Bob transferă din nou o cantitate de bani egală cu prima din contul lui Alice în al lui Trudy. La câtva timp după cea de-a 50-a pereche de mesaje pe care o primeşte, Bob aleargă afară din biroul său pentru a o găsi pe Trudy şi a-i oferi un împrumut mare astfel ca ea să-şi poată extinde afacerea ce se dovedeşte a fi atât de plină de succes. Problema se numeşte **atacul prin reluare**.

Sunt câteva soluţii posibile la atacul prin reluare. Prima este de a include în fiecare mesaj o amprentă de timp. Astfel, dacă cineva primeşte un mesaj expirat, îl ignoră. Necazul cu această abordare este că într-o reţea ceasurile nu sunt niciodată perfect sincronizate, astfel încât va exista un întreg interval de timp în care o amprentă de timp este validă . Trudy poate retrimite mesajul în acest interval de timp fără să fie prinsă .

Cea de-a doua soluţie este să se pună, în fiecare mesaj, un număr ad-hoc. Fiecare parte trebuie să-şi rememoreze toate numerele ad-hoc folosite anterior şi să respingă orice mesaj ce conţine un număr ad-hoc folosit deja. Dar numerele ad-hoc trebuie rememorate la nesfârşit, chiar şi atunci când Trudy încearcă să retrimită un mesaj vechi de 5 ani. De asemenea, dacă o maşină cade şi îşi pierde lista de numere ad-hoc, ea va fi din nou vulnerabilă la un atac prin reluare. Amprentele de timp şi numerele ad-hoc pot fi combinate pentru a limita timpul în care acestea din urmă nu trebuie şterse, dar este evident că protocolul devine mult mai complicat.

O abordare şi mai sofisticată a autentificării este folosirea unui protocol provocare-răspuns multicăi. Un exemplu binecunoscut de astfel de protocol este protocolul de **autentificare Needham- Schroeder** (Needham şi Schroeder, 1978). O variantă a acestuia este prezentată în fig. 8-40.

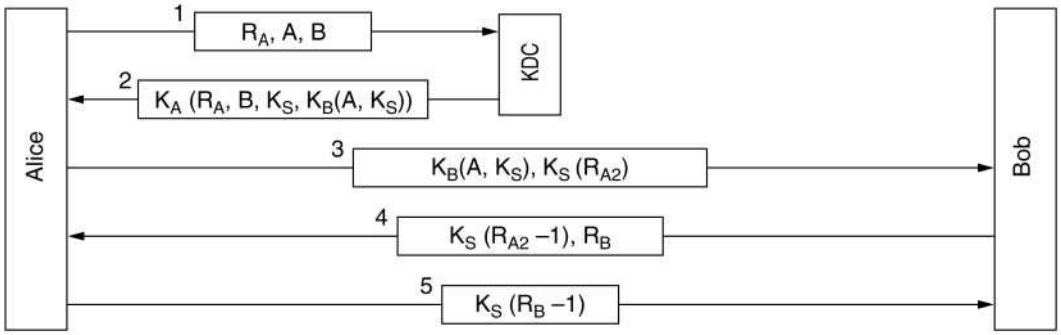


Fig. 8-40. Protocolul de autentificare Needham-Schroeder.

În acest protocol, Alice începe prin a anunţa KDC că ea doreşte să vorbească cu Bob. Acest mesaj conţine un număr aleator mare, R_A, pe post de număr ad-hoc. KDC trimite înapoi mesajul 2 conţinând numărul aleator al lui Alice împreună cu un tichet pe care ea îl poate trimite lui Bob. Scopul numărului aleator este acela de a o asigura pe Alice că mesajul 2 este proaspăt şi nu unul reluat. Identitatea lui Bob este de asemenea inclusă pentru cazul în care lui Trudy îi vine amuzanta idee să înlocuiască *B*-ul din mesajul 1 cu propria sa identitate, astfel încât KDC să cripteze tichetul de la sfârşitul mesajului 2 cu K_T în loc de K_B. Tichetul criptat cu K_B este inclus în interiorul mesajului crip- tat, pentru a o împiedica pe Trudy să-l înlocuiască cu altceva pe drumul lui înapoi spre Alice.

Acum Alice îi trimite tichetul lui Bob, împreună cu un nou număr aleator, R_{A2}, criptat cu cheia de sesiune, K_s. În mesajul 4, Bob trimite înapoi $K_S(R_2 - 1)$ pentru a-i dovedi lui Alice că vorbeşte cu adevăratul Bob. Trimiterea înapoi a lui $K_S(R_{A2})$ nu ar fi mers, deoarece ar fi fost posibil ca Trudy tocmai să-l fi furat din mesajul 3.

După primirea mesajului 4, Alice este convinsă că vorbeşte cu Bob şi că până în acest moment nu s-au folosit mesaje reluate. Doar ea tocmai generase R_{A2} cu câteva milisecunde înainte. Scopul mesajului 5 este de a-l convinge pe Bob că cea cu care vorbeşte este chiar Alice şi că nu s-au folosit nici aici mesaje reluate. Posibilitatea oricărui tip de atac prin replicare este eliminată, deoarece fiecare parte nu numai că generează o provocare, dar şi răspunde la una.

Cu toate că protocolul pare a fi destul de solid, el are o mică scăpare. Dacă Trudy reuşeşte să obţină o cheie de sesiune veche în text clar, ea poate să iniţieze o nouă sesiune cu Bob reluând mesajul 3 corespunzător cheii compromise şi convingându-l pe acesta că ea este Alice (Denning şi Sacco, 1981). De această dată ea poate prăda contul din bancă al lui Alice fără să trebuiască să se legitimeze nici măcar o dată.

Needham şi Schroeder au publicat mai târziu un protocol care corectează această problemă (Needham şi Schroeder, 1987). În acelaşi număr al aceluiaşi jurnal, Otway şi Rees (1987) au publicat de asemenea un protocol care rezolvă problema pe o cale mai scurtă. Fig. 8-41 ilustrează un protocol Otway-Rees uşor modificat.

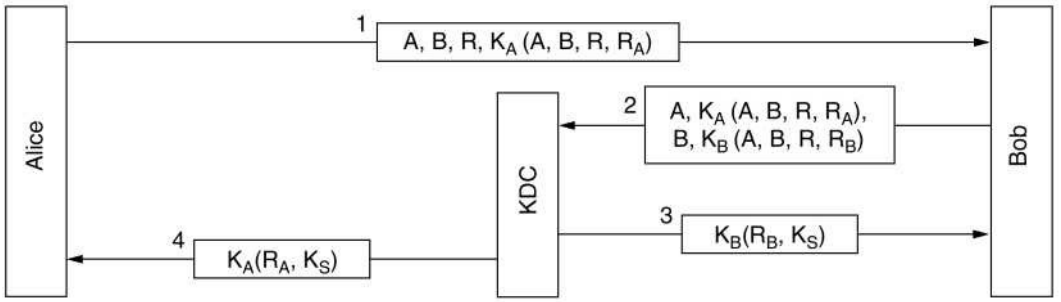


Fig. 8-41. Protocolul de autentificare Otway-Rees (puţin simplificat).

În protocolul Otway-Rees, Alice începe prin a genera o pereche de numere aleatoare, *R,* care va fi utilizată ca identificator comun, şi R_A, pe care Alice îl va folosi pentru a-l provoca pe Bob. Când Bob preia acest mesaj, el construieşte un mesaj nou din partea criptată a mesajului lui Alice şi unul analog din partea sa. Ambele părţi criptate cu K_A şi K_B, identificând pe Alice şi pe Bob, conţin identificatorul comun şi o provocare.

KDC verifică dacă *R* din ambele părţi este acelaşi. S-ar putea să nu fie, deoarece Trudy a intervenit cu *R* în mesajul 1 sau a înlocuit partea din mesajul 2. Dacă cele două R_S se potrivesc, KDC crede că mesajul de cerere de la Bob este valid. El generează atunci o cheie de sesiune şi o criptează de două ori, o dată pentru Alice şi o dată pentru Bob. Fiecare mesaj conţine numărul aleator al receptorului, ca dovadă că mesajul a fost generat de KDC şi nu de Trudy. În acest moment atât Alice cât şi Bob sunt în posesia aceleiaşi chei de sesiune şi pot începe comunicarea. Prima dată când ei vor schimba mesaje de date, fiecare va putea vedea că celălalt are o copie identică a lui KS, astfel autentificarea fiind completă.

8.7.4 Autentificarea folosind Kerberos

Un protocol de autentificare folosit în multe sisteme reale (inclusiv Windows 2000) este **Kerberos**, care se bazează pe o variantă a protocolului Needham-Schroeder. Numele său vine de la un câine cu mai multe capete din mitologia greacă, ce era folosit pentru a păzi intrarea în Hades (probabil pentru a-i ţine pe cei nedoriţi afară). Kerberos a fost proiectat la M.I.T. pentru a permite utilizatorilor de la staţiile de lucru să acceseze resursele reţelei într-un mod sigur. Cea mai mare diferenţă dintre el şi Needham-Schroeder este presupunerea lui că toate ceasurile sunt destul de bine sincronizate. Protocolul a trecut prin câteva iteraţii. V4 este versiunea cea mai larg utilizată în industrie, aşa că pe aceasta o vom descrie. După care vom spune câteva cuvinte despre succesoarea sa, V5. Pentru mai multe informaţii, vezi (Steiner et. al, 1988).

Kerberos implică trei servere în afară de Alice (staţia de lucru a clientului):

Serverul de autentificare (AS – eng.: *Authentication Server):* verifică utilizatorii în timpul conectării.

Serverul de acordare a Tichetelor (TGS – eng.: *Ticket-Granting Server):* emite „demonstrarea identităţii tichetelor”.

Serverul Bob: realizează efectiv acţiunea pe care o doreşte Alice

AS este similar unui KDC prin aceea că el partajează o parolă secretă cu orice utilizator. Sarcina TGS este de a emite tichete care pot convinge serverele reale că acela care deţine un tichet TGS este într-adevăr cel ce pretinde a fi.

Pentru a porni o sesiune, Alice stă aşezată la o staţie de lucru publică oarecare şi-şi tastează numele. Staţia de lucru transmite numele ei la AS ca text clar, după cum este arătat în fig. 8-42. Ceea ce vine înapoi este o cheie de sesiune împreună cu un tichet $K_{TGS}(A, K_S)$, destinate TGS-ului. Aceste elemente sunt împachetate împreună şi criptate folosind cheia secretă a lui Alice, astfel încât doar Alice să le poată decripta. Doar când soseşte mesajul 2, staţia de lucru îi va cere lui Alice parola. Parola este folosită pentru a genera K_A , în scopul decriptării mesajului 2 şi obţinerii cheii de sesiune şi tichetului TGS din interiorul acestui mesaj. În acest moment, staţia de lucru înlocuieşte parola lui Alice pentru a se asigura că ea se găseşte în interiorul staţiei de lucru pentru cel mult câteva milise- cunde. Dacă Trudy încearcă să se conecteze ca Alice, parola pe care o introduce va fi greşită şi staţia de lucru va detecta acest lucru deoarece partea standard a mesajului 2 va fi incorectă.

După conectare Alice trebuie să anunţe staţia de lucru că doreşte să-l contacteze pe Bob, serverul de fişiere. Atunci staţia de lucru trimite spre TGS mesajul 3 cerând un tichet pentru a-l folosi cu Bob. Elementul cheie în această cerere este $K_{TGS}(A, K_S)$, care este criptat cu cheia secretă a TGS şi este folosit ca dovadă că transmiţătorul este chiar Alice. TGS răspunde prin crearea unei chei de sesiune K_B pe care Alice să o folosească cu Bob. Se trimit înapoi două versiuni ale acesteia. Prima este criptată doar cu K_S, astfel încât Alice să poată să o citească. A doua este criptată cu cheia lui Bob, K_B, astfel încât Bob să o poată citi.

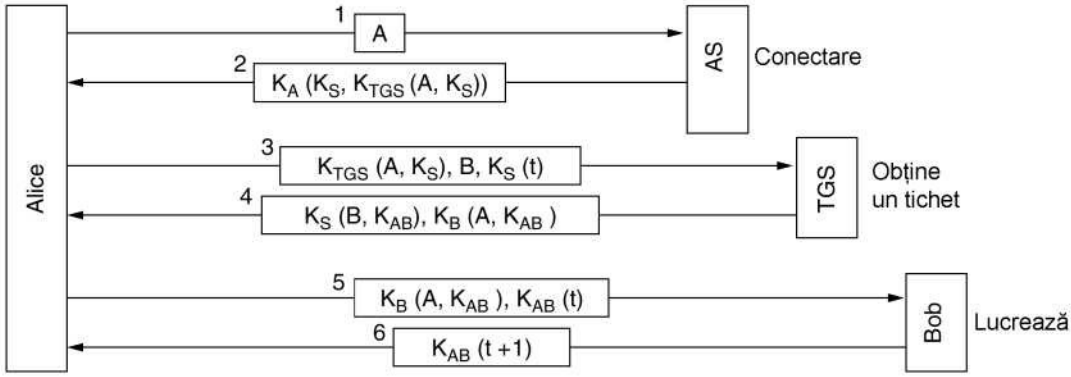


Fig. 8-42. Modul de operare la Kerberos V4.

Trudy poate copia mesajul 3 şi poate încerca să-l utilizeze din nou, dar ea va fi împiedicată de amprenta de timp criptată, *t*, trimisă împreună cu el. Trudy nu poate înlocui amprenta de timp cu una mai recentă, deoarece ea nu cunoaşte K_S, cheia de sesiune pe care o foloseşte Alice pentru a vorbi cu TGS. Chiar dacă Trudy retrimite mesajul 3 repede, tot ceea ce va obţine este o altă copie a mesajului 4, pe care nu a putut să-l decripteze prima dată şi nu va putea nici a doua oară.

Acum Alice îi poate trimite lui Bob K_{AB} pentru a stabili o sesiune cu el. Acest schimb este de asemenea cu amprente de timp. Răspunsul este pentru Alice dovada

că ea vorbeşte chiar cu Bob, nu cu Trudy.

După această serie de schimburi, Alice poate comunica cu Bob sub acoperirea cheii K_{AB}. Dacă mai târziu se decide că are nevoie să comunice cu alt server, Carol, ea va repeta doar mesajul 3 spre TGS, specificând de această dată *C* în loc de *B*. TGS va răspunde cu promptitudine cu un tichet criptat cu K_C, pe care Alice îl poate transmite lui Carol şi pe care Carol îl va accepta ca dovadă că el vine de la Alice.

Scopul întregii acţiuni este că acum Alice poate accesa serverele din toată reţeaua într-un mod sigur şi că parola sa nu va fi niciodată transmisă prin reţea. De fapt parola trebuie să existe pe staţia de lucru doar pentru câteva milisecunde. Cu toate acestea, trebuie remarcat că fiecare server face propria sa autorizare. Atunci când Alice îi prezintă tichetul său lui Bob, acest tichet doar îi demonstrează lui Bob cine l-a trimis. Ceea ce îi este permis lui Alice să facă, depinde doar de Bob.

Deoarece proiectanţii Kerberos-ului nu s-au aşteptat ca întreaga lume să aibă încredere într-un singur server de autentificare, ei au prevăzut posibilitatea de a avea **domenii** multiple, fiecare cu propriul său AS şi TGS. Pentru a obţine un tichet de la un server dintr-un domeniu de la distanţă, Alice ar trebui să ceară propriului său TGS un tichet acceptat de TGS-ul din domeniul de la distanţă. Dacă TGS-ul de la distanţă este înregistrat la TGS-ul local (în acelaşi mod în care sunt serverele locale), TGS-ul local îi va da lui Alice un tichet valid pentru TGS-ul de la distanţă. Astfel ea poate să lucreze acolo, de exemplu să ia tichete pentru serverele din acest domeniu. De notat totuşi că, pentru ca părţi din două domenii diferite să conlucreze, fiecare trebuie să se încreadă în TGS-ul celuilalt.

Kerberos V5 este mai sofisticat decât V4 şi are o supraîncărcare mai mare. Pentru a descrie tipurile de date el foloseşte OSI ASN.1 (Abstract Syntax Notation 1) şi prezintă mici modificări în protocoale. Mai mult decât atât, are timpi de viaţă mai mari pentru tichete, permite reînnoirea tichetelor şi va elibera tichete postdatate. În plus, cel puţin în teorie, nu este dependent de DES, cum este V4, şi suportă domenii multiple delegând servere de tichete multiple pentru generarea de tichete.

8.7.5 Autentificarea folosind criptografia cu cheie publică

Autentificarea mutuală poate fi realizată şi cu ajutorul criptografiei cu cheie publică. Pentru început, Alice are nevoie de cheia publică a lui Bob. Dacă există o PKI cu un server de directoare care emite certificate pentru chei publice, Alice o poate cere pe a lui Bob, după cum se vede din fig. 8-43, ca mesajul 1. Răspunsul, în mesajul 2, este un certificat X.509 conţinând cheia publică a lui Bob. Când Alice verifică faptul că semnătura e corectă, îi trimite lui Bob un mesaj conţinând identitatea ei şi un număr ad-hoc.

Când Bob primeşte acest mesaj, nu are idee dacă a venit de la Alice sau de la Trudy, dar continuă şi cere serverului de directoare cheia publică a lui Alice (mesajul 4), pe care o obţine imediat (mesajul 5). Apoi îi trimite lui Alice un mesaj conţinând R_A a lui Alice, propriul număr ad-hoc, R_B, şi o cheie de sesiune propusă, K_S, ca mesajul 6.

Când Alice primeşte mesajul 6, ea îl decriptează folosind propria cheie privată. Ea vede R_A în mesaj, ceea ce îi dă o senzaţie plăcută. Mesajul trebuie să fi venit de

la Bob, pentru că Trudy nu are nici un mijloc de a determina R_A. Mai mult, trebuie să fie un mesaj proaspăt, şi nu unul reluat, din moment ce ea tocmai i-a trimis R_A lui Bob. Alice acceptă cheia de sesiune trimiţând înapoi mesajul 7. Când Bob vede R_B criptat cu cheia de sesiune pe care a generat-o el, ştie că Alice a primit mesajul 6 şi a verificat R_A.

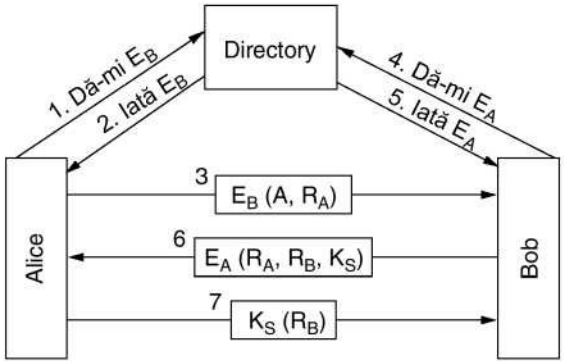


Fig. 8-43. Autentificarea mutuală folosind criptografia cu cheie publică.

Cum poate încerca Trudy să compromită acest protocol? Ea poate fabrica mesajul 3 şi-l poate păcăli pe Bob să o testeze pe Alice, dar Alice va vedea un R_A pe care nu l-a trimis ea şi nu va merge niciodată mai departe. Trudy nu poate falsifica mesajul 7 pentru Bob, deoarece nu ştie R_B sau K_S şi nu le poate determina fără cheia privată a lui Alice. De data aceasta ea nu are noroc.

8.8 CONFIDENŢIALITATEA POŞTEI ELECTRONICE

Când un mesaj de poştă electronică este trimis între două locaţii situate la distanţă, acesta va trece în drumul său pe la un număr mare de maşini. Oricare dintre acestea poate să citească şi să înregistreze mesajul pentru utilizare ulterioară. Confidenţialitatea este inexistentă, în ciuda a ceea ce mulţi oameni cred (Weisband şi Reining, 1995). Cu toate acestea, multora le-ar plăcea să poată trimite mesaje care să fie citite doar de cei cărora le sunt adresate şi de nimeni altcineva: nu de şeful lor şi nici chiar de guvern. Această dorinţă a stimulat mai multe persoane şi grupuri să aplice principiile criptografiei, studiate de noi anterior, la poşta electronică, pentru a produce un sistem de e-mail sigur. În secţiunile următoare vom studia un sistem de e-mail sigur larg utilizat, PGP, şi apoi vom menţiona pe scurt altele două, PEM şi S/MIME. Pentru informaţii suplimentare a se vedea (Kaufman et. al, 2002; şi Schneier, 1995).

8.8.1 PGP-Pretty Good Privacy (rom.: Confidenţialitate Destul de Bună)

Primul nostru exemplu, **PGP** (eng.: **Pretty Good Privacy**) este în cea mai mare parte rodul gândirii unei singure persoane, Phil Zimmermann (Zimmermann, 1995a, 1995b). Zimmermann este un avocat al confidenţialităţii, al cărui motto este: Dacă dreptul la confidenţialitate este în afara legii, atunci doar cei aflaţi în afara legii vor avea confidenţialitate. Lansat în 1991, PGP este un pachet complet de securitate a e-mail-ului, care oferă confidenţialitate, autentificare, semnături digitale şi compresie, toate într-o formă uşor de utilizat. Pe deasupra, pachetul complet, incluzând tot codul
sursă, este distribuit gratuit prin Internet. Datorită calităţii sale, preţului (zero) şi disponibilităţii pe platformele UNIX, Linux, Windows şi Mac OS, acesta este larg utilizat astăzi.

PGP criptează datele folosind un cifru bloc numit **IDEA** (eng.: **International Data Encryption Algorithm**, rom.: **Algoritm Internaţional de Criptare a Datelor**), care foloseşte chei de 128 de biţi. Acesta a fost proiectat în Elveţia într-un moment în care DES era considerat compromis şi AES încă nu fusese inventat. Conceptual, IDEA este similar cu DES şi AES: permută biţii într-un şir de runde, dar detaliile funcţiilor de permutare sunt diferite de DES şi AES. Gestiunea cheilor foloseşte RSA şi integritatea datelor foloseşte MD5, subiecte pe care le-am discutat deja.

PGP a fost implicat în diferite controverse începând din prima zi (Levy, 1993). Deoarece Zimmermann nu a făcut nimic pentru a împiedica alte persoane să distribuie PGP pe Internet, de unde poate fi luat de oameni din toată lumea, guvernul Statelor Unite a pretins că Zimmermann a violat legile privind exportul de muniţie. Investigarea lui Zimmermann de către guvernul Statelor Unite a durat 5 ani, dar până la urmă a încetat, probabil din două motive. În primul rând, Zimmermann nu a pus el însuşi PGP pe Internet, aşa că avocatul lui a susţinut că *el* nu a exportat niciodată nimic (şi apoi mai este şi problema dacă de fapt a crea un sit Web constituie un export). În al doilea rând, guvernul şi-a dat seama în cele din urmă că a câştiga un proces înseamnă a convinge juraţii că un sit Web conţinând un program de confidenţialitate ce poate fi descărcat intră sub incidenţa legii traficului de arme, care interzice exportul de material de război cum ar fi tancurile, submarinele, aparatele de zbor militare şi armele nucleare. Şi probabil că nici anii întregi de publicitate negativă nu au ajutat prea mult.

Făcând o paranteză, legile exportului sunt bizare, ca să folosim termeni moderaţi. Guvernul a considerat punerea de cod pe un sit Web ilegală şi l-a hărţuit pe Zimmermann 5 ani pentru acest lucru. Pe de altă parte, când cineva a publicat codul sursă complet al PGP, în C, într-o carte (cu caractere mari şi cu sume de control pe fiecare pagină, ca să poată fi verificat uşor) şi apoi a exportat cartea, nu au fost probleme cu guvernul deoarece cărţile nu sunt clasificate ca muniţii. Sabia este mai puternică decât condeiul, cel puţin pentru Unchiul Sam.

O altă problemă cu care s-a confruntat PGP a fost una de încălcare a patentului. Compania ce deţine patentul RSA, RSA Security, Inc., a susţinut că folosirea de către PGP a algoritmului RSA îi încalcă patentul, dar această problemă a fost rezolvată în versiunile începând de la 2.6. Mai mult, PGP foloseşte şi alt algoritm de criptare patentat, IDEA, ceea ce a provocat nişte probleme la început.

Cum sursele PGP sunt publice, diverse persoane şi grupuri l-au modificat şi au produs un număr de versiuni. Unele au fost proiectate ca să ocolească problemele cu legile muniţiei, altele s-au orientat spre evitarea folosirii algoritmilor patentaţi, şi altele au vrut să-l transforme într-un produs comercial, în care sursele să nu mai fie disponibile. Deşi legile muniţiei au mai fost liberalizate puţin (altfel produsele ce folosesc AES nu ar fi putut fi exportate din S.U.A.) şi patentul RSA a expirat în septembrie 2000, consecinţa tuturor acestor probleme este că acum sunt în circulaţie diverse versiuni incompatibile ale PGP, sub nume diferite. Discuţia de mai jos se referă la PGP clasic, adică cea mai veche şi mai simplă versiune. Altă versiune populară, Open PGP, este descrisă în RFC 2440. Încă o altă versiune este GNU Privacy Guard.

În mod intenţionat PGP utilizează algoritmi de criptare existenţi, în loc să inventeze unii noi. Se bazează în principal pe algoritmi care au fost supuşi unor numeroase analize amănunţite şi nu au fost proiectaţi sau influenţaţi de vreo agenţie guvernamentală care să încerce să le micşoreze puterea. Pentru cei ce au tendinţa să nu aibă încredere în guvern, această proprietate este un mare avantaj.

PGP permite compresia de text, asigurarea secretului mesajelor şi semnături digitale şi furnizează de asemenea facilităţi de management extensiv al cheilor, dar, destul de ciudat, nu şi facilităţi pen

tru e-mail. Este mai mult un preprocesor care preia text clar de la intrare şi produce text semnat, cifrat, în bază 64, ca rezultat. Acest rezultat poate fi apoi trimis prin e-mail, bineînţeles. Unele implementări ale PGP apelează, ca pas final, un agent utilizator care să trimită efectiv mesajul.

Pentru a vedea cum funcţionează PGP, să considerăm exemplul din fig. 8-44. Aici, Alice vrea să-i trimită lui Bob, într-o manieră sigură, un mesaj text simplu, semnat, P. Alice şi Bob au cheile private *(D_X)* şi cheile RSA publice *(E_X).* Să presupunem că fiecare ştie cheia publică a celuilalt; ne vom ocupa mai târziu de administrarea cheilor.

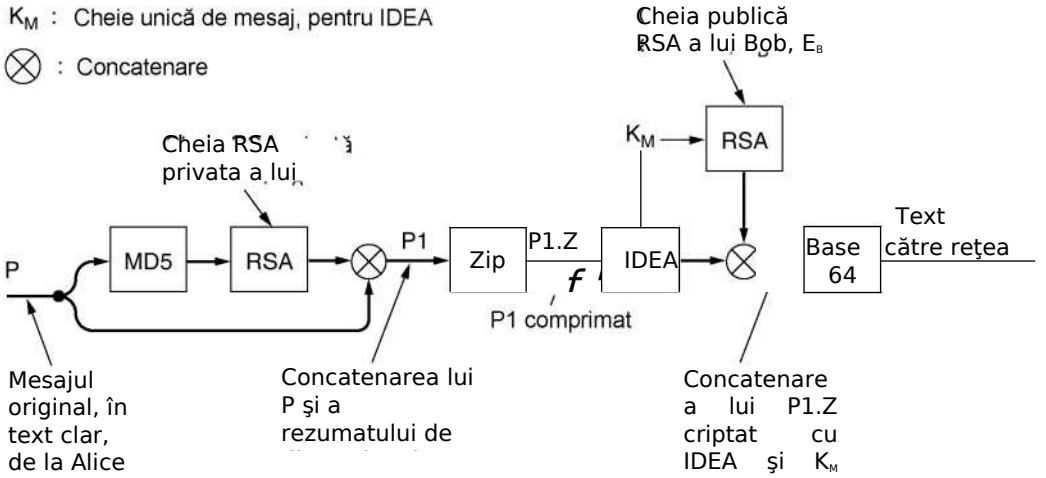


Fig. 8-44. PGP în acţiunea de trimitere a unui mesaj.

Alice începe prin a invoca programul PGP pe calculatorul său. Mai întâi PGP rezumă prin dispersare (eng.: hash) mesajul P utilizând MD5, apoi criptează codul de dispersie rezultat utilizând cheia sa RSA privată, *DA.* Când, în cele din urmă, Bob primeşte mesajul, el poate decripta rezumatul cu cheia publică cunoscută a lui Alice şi poate testa corectitudinea acestuia. Chiar dacă altcineva (de ex. Trudy) poate obţine codul în această etapă şi îl poate decripta cu cheia publică a lui Alice, puterea lui MD5 garantează că este nerealizabilă computaţional producerea unui alt mesaj care să aibă acelaşi cod MD5.

Rezumatul criptat şi mesajul original sunt acum concatenate într-un singur mesaj *P1* şi comprimate apoi cu programul ZIP, care utilizează algoritmul Ziv-Lempel (Ziv şi Lempel, 1977). Numim ieşirea obţinută la acest pas *P1.Z.*

Apoi, PGP îi cere lui Alice introducerea unui şir de caractere oarecare. Atât conţinutul acestuia cât şi viteza de tastare sunt utilizate pentru a genera o cheie de mesaj de tip IDEA, de 128 de biţi, K_M (numită cheie de sesiune în literatura PGP, dar numele este nepotrivit atâta timp cât nu există nici o sesiune). K_M este acum utilizat pentru a cripta *P1.Z* cu IDEA, prin metoda de tip reacţie cifrată. În plus, K_M este criptată cu cheia publică a lui Bob, E_B. Aceste două componente sunt apoi concatenate şi convertite în bază 64, aşa cum s-a discutat în secţiunea despre

MIME din Cap. 7. Mesajul rezultat conţine numai litere, cifre şi simbolurile +, / şi =, ceea ce înseamnă că poate fi pus într-un corp de RFC 822 şi că ne putem aştepta să ajungă nemodificat la destinaţie.

Când Bob primeşte mesajul, îl reconverteşte din bază 64 şi decriptează cheia IDEA utilizând cheia sa RSA privată. Utilizând această cheie, decriptează mesajul pentru a obţine *P1.Z.* După de-

compresia acestuia, Bob separă textul simplu de codul cifrat şi decriptează codul de dispersie utilizând cheia publică a lui Alice. Dacă codul textului clar coincide cu ceea ce a calculat el utilizând MD5, el ştie că P este mesajul corect şi că provine de la Alice.

Merită observat că RSA este utilizat doar în două locuri aici: pentru a cifra codul de dispersie de 128 de biţi generat de MD5 şi pentru a cifra cheia IDEA de 128 de biţi. Deşi RSA este lent, are de criptat doar 256 de biţi şi nu un volum mare de date. Mai mult, toţi cei 256 de biţi de text simplu sunt generaţi extrem de aleator, astfel încât numai pentru a determina dacă o cheie ghicită este corectă, Trudy ar trebui să depună o cantitate însemnată de muncă. Criptarea de mare putere este realizată de IDEA, care este cu câteva ordine de mărime mai rapidă decât RSA. Astfel, PGP asigură securitate, compresie şi o semnătură digitală şi face acest lucru într-o manieră chiar mult mai eficientă decât schema ilustrată în fig. 8-19.

Obişnuită (384 biţi): poate fi spartă uşor în ziua de azi.
Comercială (512 biţi): ar putea fi spartă de organizaţii cu nume din trei litere (care se ocupă cu securitatea statului).

3. Militară (1024 biţi): Nu poate fi spartă de nici un pământean.
4. Extraterestră (2048 biţi): Nu poate fi spartă nici de cineva de pe altă planetă.

Cum RSA este folosită doar pentru două calcule uşoare, toată lumea ar trebui să folosească mereu chei de lungime extraterestră.

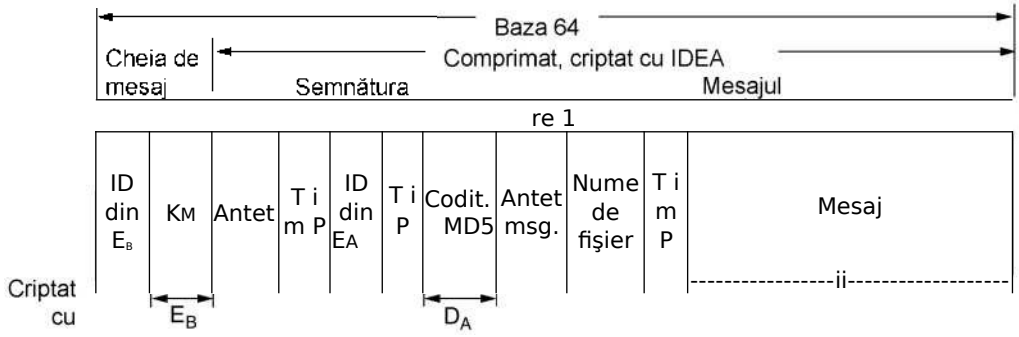


Fig. 8-45. Un mesaj

PGP acceptă trei lungimi de chei RSA. Rămâne la latitudinea utilizatorului să o aleagă pe cea mai potrivită. Lungimile disponibile sunt:

1.
2.

Formatul unui mesaj PGP este prezentat în fig. 8-45. Mesajul are trei părţi, conţinând cheia IDEA, semnătura şi respectiv mesajul. Partea care conţine cheia mai include, de asemenea, un identificator de cheie, deoarece utilizatorilor li se permite să aibă mai multe chei publice.

Partea cu semnătura conţine un antet, de care nu ne vom ocupa aici. Antetul este urmat de: o amprentă de timp, identificatorul pentru cheia publică a emiţătorului, care poate fi folosit pentru a decripta codul semnăturii, unele informaţii care identifică algoritmul folosit (pentru a permite folosirea algoritmilor

MD6 şi RSA2 atunci când aceştia vor fi inventaţi) precum şi codul de dispersie criptat.

Partea de mesaj conţine de asemenea un antet, numele implicit care va fi folosit pentru fişier în cazul în care utilizatorul doreşte să-l salveze pe disc, o amprentă de timp pusă la crearea mesajului şi, în sfârşit, mesajul în sine.

Administrarea cheilor a fost tratată cu atenţie foarte mare în PGP, deoarece acesta este călcâiul lui Ahile pentru sistemele de securitate. Administrarea cheilor funcţionează după cum urmează. Fiecare utilizator menţine două structuri de date locale: un inel cu chei private şi unul cu chei publice. **Inelul cheilor private** conţine una sau mai multe perechi de chei personale (privată-publică). Motivul pentru care se acceptă mai multe chei pentru un utilizator este pentru a permite acestora să-şi schimbe cheile publice periodic sau când se consideră că acestea au fost compromise, fără a invalida mesajele aflate în pregătire sau în tranzit. Fiecare pereche are un identificator asociat, astfel încât un emiţător de mesaj poate spune destinatarului ce cheie publică a fost folosită pentru criptarea acestuia. Identificatorii de mesaj constau din ultimii 64 de biţi ai cheii publice. Utilizatorii sunt responsabili pentru evitarea conflictelor între identificatorii cheilor lor publice. Cheile private de pe disc sunt criptate folosind o parolă specială (arbitrar de lungă) pentru a le proteja împotriva unor atacuri.

Inelul cheilor publice conţine cheile publice ale corespondenţilor utilizatorului. Acestea sunt necesare pentru criptarea cheilor de mesaj asociate cu fiecare mesaj. Fiecare intrare în inelul cheilor publice conţine nu doar cheia publică, ci şi identificatorul său pe 64 de biţi şi o indicaţie asupra încrederii pe care o are utilizatorul în cheie.

Problema care trebuie rezolvată aici este următoarea. Să presupunem că cheile publice sunt menţinute în grupurile de ştiri. O metodă pentru Trudy de a citi e-mail-ul secret al lui Bob este de a ataca grupul de ştiri şi de a înlocui cheia publică a lui Bob cu una la alegerea sa. Când Alice va lua mai târziu aşa-zisa cheie a lui Bob, Trudy poate lansa un atac găleata-brigăzii la adresa lui Bob.

Pentru a preveni astfel de atacuri, sau cel puţin pentru a minimiza consecinţele lor, Alice trebuie să ştie cât de mult se poate încrede în obiectul numit „cheia lui Bob” din inelul său de chei publice. Dacă ea ştie că Bob personal i-a dat o dischetă conţinând cheia, atunci poate acorda valoarea maximă de încredere. Această abordare descentralizată, bazată pe controlul utilizatorului, asupra administrării cheilor publice este ceea ce face PGP să se deosebească de schemele centralizate PKI.

Totuşi, în practică, oamenii primesc cheile publice întrebând un server de chei sigur. Din acest motiv, după ce X.509 a fost standardizat, PGP a început să suporte aceste certificate, pe lângă tradiţionalul mecanism al inelelor cu chei publice. Toate versiunile curente de PGP suportă X.509.

8.8.2 PEM-Privacy Enhanced Mail (Poştă cu Confidenţialitate Sporită)

Spre deosebire de PGP, care a fost iniţial opera unui singur om, cel de-al doilea exemplu al nostru, **PEM (**eng.: **Privacy Enhanced Mail)**, dezvoltat la sfârşitul anilor '80, este un standard oficial Internet şi este descris în patru RFC-uri: RFC 1421

până la RFC 1424. Foarte pe scurt, PEM acoperă acelaşi teritoriu ca şi PGP: confidenţialitatea şi autentificarea sistemelor de e-mail bazate pe RFC 822. Cu toate acestea, el prezintă şi unele diferenţe faţă de abordarea şi tehnologia PGP.

Mesajele trimise folosind PEM sunt mai întâi convertite într-o formă canonică, astfel încât ele au aceleaşi convenţii referitoare la spaţii albe (de ex. tab-uri, spaţiile de la sfârşit de text). Apoi este calculat un cod de dispersie al mesajului, folosind MD2 sau MD5. După aceasta, concatenarea codului de dispersie şi a mesajului este criptată folosind DES. În lumina cunoscutei slăbiciuni a unei chei de 56 de biţi, această alegere este în mod sigur suspectă. Mesajul criptat poate fi apoi codificat utilizând o codificare în baza 64 şi transmis destinatarului.

Ca şi în PGP, fiecare mesaj este criptat cu o cheie unică, inclusă şi ea în mesaj. Cheia poate fi protejată fie cu RSA, fie cu DES triplu folosind EDE.

În PEM administrarea cheilor este mult mai structurată decât în PGP. Cheile sunt certificate prin certificate X.509 emise de CA-uri, care sunt organizate într-o ierarhie rigidă pornind de la o

singură rădăcină (bază). Avantajul acestei scheme este că revocarea certificatului este posibilă dacă autoritatea corespunzătoare rădăcinii emite periodic CRL-uri.

Singura problemă cu PEM este că nimeni nu l-a folosit niciodată şi a dispărut de mult în neant. În mare, problema a fost politică: cine ar funcţiona ca bază a ierarhiei şi în ce condiţii? Nu a fost lipsă de candidaţi, dar multora le-a fost frică să încredinţeze vreuneia dintre companii securitatea întregului sistem. Cel mai serios candidat, RSA Security, Inc., a vrut să perceapă o taxă pentru fiecare certificat emis. Totuşi, unele organizaţii au respins ideea. În particular, guvernul Statelor Unite poate folosi toate patentele din ţară fără a plăti, iar companiile din afara S.U.A. se obişnuiseră să folosească algoritmul RSA pe gratis (compania a uitat să îl patenteze în afara Statelor Unite). Nici unii, nici alţii nu erau entuziasmaţi de ideea de a trebui deodată să plătească RSA Security, Inc. pentru lucruri pe care ei le făcuseră dintotdeauna pe gratis. Până la urmă, nu a putut fi găsită o bază pentru ierarhie şi PEM s-a prăbuşit.

8.8.3 S/MIME

Următoarea aventură a IETF în domeniul securităţii poştei electronice, denumită **S/MIME** (**Se- cure/MIME,** rom.: **MIME Sigur**), este descrisă în RFC 2632 până la 2643. Ca şi PEM, acesta oferă autentificare, integritatea datelor, confidenţialitate şi non-repudiere. Este de asemenea destul de flexibil, suportând o varietate de algoritmi criptografici. În mod previzibil, având în vedere numele, S/MIME se integrează bine cu MIME, permiţând tuturor tipurilor de mesaje să fie protejate. Sunt definite o varietate de antete noi MIME, de exemplu pentru a conţine semnături digitale.

IETF a învăţat evident ceva din experienţa cu PEM. S/MIME nu are o ierarhie rigidă pentru certificate, pornind de la o singură bază. În schimb, utilizatorii pot avea mai multe puncte de încredere (eng.: trust anchors). Atâta timp cât un certificat poate fi urmărit înapoi până la un punct în care utilizatorul are încredere, este considerat valid. S/MIME foloseşte algoritmii şi protocoalele standard pe care

le-am examinat până acum, deci nu îl vom mai discuta aici. Pentru detalii, vă rog să consultaţi RFC-urile.

8.9 SECURITATEA WEB-ULUI

Am studiat două domenii importante în care securitatea este necesară: comunicaţiile şi poşta electronică. Puteţi să vi le imaginaţi pe acestea ca fiind supa şi aperitivul. Acum este momentul pentru felul principal: securitatea Web-ului. Web-ul este locul unde în zilele noastre cele mai multe Trudy îşi petrec vremea făcând răutăţi. În secţiunile următoare vom analiza unele probleme şi aspecte referitoare la securitatea Web-ului.

În linii mari, securitatea Web-ului poate fi împărţită în trei subiecte. În primul rând, cum pot fi sigur denumite obiectele şi resursele? În al doilea rând, cum se pot stabili conexiuni sigure, autentificate? În al treilea rând, ce se întâmplă când un sit Web trimite unui client un fragment de cod executabil? După ce prezentăm câteva tipuri de pericole, vom examina toate aceste probleme.

8.9.1 Pericole

În ziare se poate citi despre problemele de securitate a siturilor Web aproape în fiecare săptămână. Situaţia e într-adevăr destul de proastă. Să vedem câteva exemple de lucruri care s-au întâmplat deja. În primul rând, paginile principale ale multor organizaţii au fost atacate şi înlocuite cu alte pagini alese de spărgători (eng.: crackers). (Presa populară denumeşte persoanele care sparg sistemele „hackers”, dar mulţi programatori rezervă acest termen pentru marii programatori. Preferăm să denumim aceste persoane „spărgători” – „crackers”). Printre siturile care au fost sparte se numără şi Yahoo, Armata S.U.A., CIA, NASA, New York Times. În cele mai multe cazuri, spărgătorii au pus acolo doar un text amuzant şi siturile au fost reparate în câteva ore.

Acum să vedem nişte cazuri mult mai serioase. Multe situri au căzut din cauza atacurilor de refuz al serviciilor (eng.: denial-of-service attacks), în care spărgătorul „inundă” situl cu trafic, făcându-l incapabil de a răspunde la cererile legitime. De multe ori atacul e lansat de pe un număr mare de maşini în care spărgătorul tocmai a intrat (atacuri distribuite de negare a serviciilor – DDoS attacks). Aceste atacuri au devenit atât de obişnuite încât acum nici nu mai ajung la ştiri, dar pot costa situl atacat mii de dolari în afaceri pierdute.

În 1999, un spărgător suedez a intrat în situl Hotmail al firmei Microsoft şi a creat un sit oglindă (mirror) care permitea oricui să scrie numele unui utilizator Hotmail şi apoi să citească toată poşta electronică, cea curentă şi cea arhivată, a persoanei respective.

În alt caz, un spărgător rus de 19 ani numit Maxim a intrat într-un sit de comerţ electronic şi a furat 300000 de numere de cărţi de credit. Apoi i-a abordat pe deţinătorii sitului şi le-a spus că dacă nu îl plătesc cu 100000$, va publica toate numerele de cărţi de credit pe Internet. Ei nu au cedat şantajului şi el chiar a

publicat numerele cărţilor de credit, producând mari pagube multor victime inocente.

În alt stil, un student de 23 de ani din California a trimis prin e-mail unei agenţii de presă un comunicat conţinând afirmaţia falsă că Emulex, o corporaţie americană, va suferi mari pierderi şi că directorul executiv îşi va da imediat demisia. În câteva ore, acţiunile companiei au scăzut cu 60%, provocând celor care le deţineau pierderi de peste 2 miliarde $. Autorul faptei a câştigat un sfert de milion de dolari vânzând acţiunile cu puţin timp înainte de a trimite anunţul. Chiar dacă acest eveniment nu a fost o spargere a unui sit Web, este clar că punerea unui astfel de anunţ pe pagina unei mari corporaţii ar avea un efect similar.

Am putea (din nefericire) să continuăm aşa pe multe pagini. Dar acum este timpul să examinăm unele aspecte tehnice legate de securitatea Web-ului. Pentru mai multe informaţii despre probleme de securitate de orice fel, vezi (Anderson, 2001; Garfinkel şi Spafford, 2002; şi Schneier, 2000). Şi căutând pe Internet se va obţine un număr imens de astfel de cazuri.

8.9.2 Siguranţa numelor

Să începem cu ceva simplu: Alice vrea să viziteze situl Web al lui Bob. Ea tastează URL-ul lui Bob în browser şi după câteva secunde, apare o pagină Web. Dar este pagina lui Bob? Pate că da, poate că nu. Poate că Trudy s-a întors din nou la farsele ei. De exemplu, ar putea să intercepteze toate pachetele ce vin de la Alice şi să le examineze. Când găseşte o cerere HIT'P de tip *GET* pentru situl lui Bob, ar putea să se ducă ea însăşi la situl lui Bob ca să ia pagina, să o modifice după cum doreşte şi să îi trimită lui Alice pagina falsă. Alice nu ar fi deloc mai înţeleaptă. Mai rău, Trudy ar putea să micşoreze preţurile din magazinul virtual al lui Bob pentru a face produsele să pară foarte atractive, păcălind-o astfel pe Alice să trimită numărul cărţii ei de credit lui „Bob” ca să cumpere ceva.

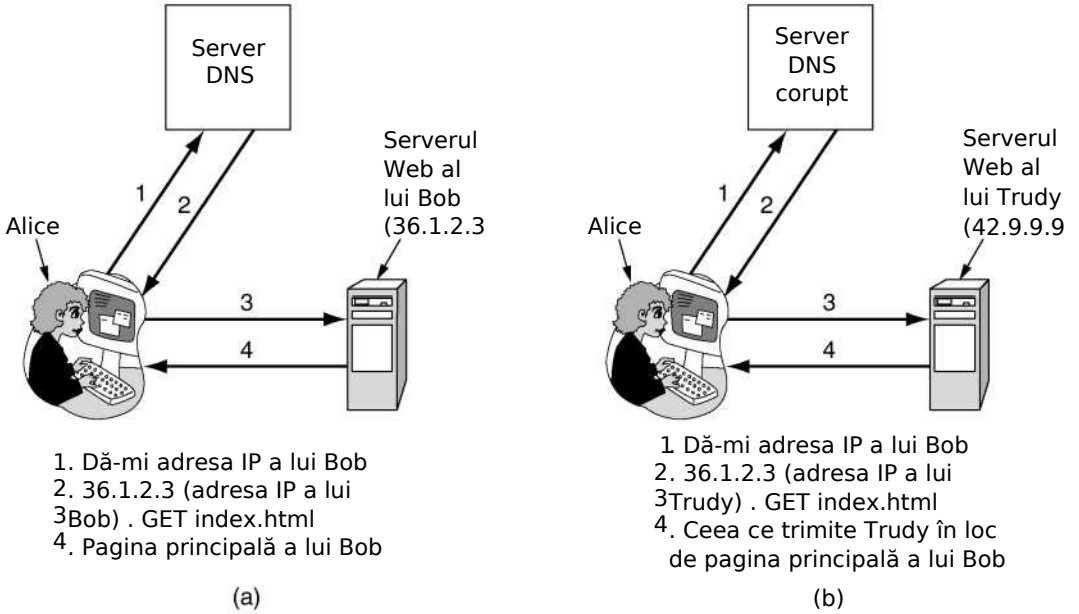


Fig. 8-46. (a) Situaţie normală. (b) Atac bazat pe spargerea DNS-ului şi modificarea înregistrării lui Bob.

Un dezavantaj al acestui atac clasic de tip omul-din-mijloc este acela că Trudy trebuie să poată intercepta traficul ce pleacă de la Alice şi să îl poată falsifica pe cel ce vine. Practic, ea trebuie să intercepteze linia de telefon a lui Alice sau a lui Bob, pentru că interceptarea fibrei optice este foarte dificilă. Deşi interceptarea activă a cablurilor este cu siguranţă posibilă, ea presupune o anumită cantitate de muncă, iar Trudy, cu toate că este inteligentă, este leneşă. În plus, există moduri mai uşoare de a o păcăli pe Alice.

Păcălirea DNS-ului (DNS Spoofing)

De exemplu, să presupunem că Trudy poate să spargă sistemul DNS, sau poate memoria ascunsă a DNS de la ISP-ul lui Alice, şi să înlocuiască adresa IP a lui Bob (să zicem, 36.1.2.3) cu adresa ei (a lui Trudy, să zicem, 42.9.9.9). Aceasta duce la următorul atac. Modul în care ar trebui să funcţioneze este ilustrat în fig. 8-46(a). Aici (1) Alice cere DNS-ului adresa IP a lui Bob, (2) o obţine, (3) îi cere lui Bob pagina sa principală şi (4) o obţine şi pe aceasta. După ce Trudy a modificat înregistrarea DNS corespunzătoare lui Bob astfel încât aceasta să conţină adresa ei IP în loc de cea a lui Bob, ajungem la situaţia din fig. 8-46(b). Aici, când Alice caută adresa IP a lui Bob, o obţine pe a lui Trudy, aşa că tot traficul ei destinat lui Bob ajunge la Trudy. Trudy poate organiza acum un atac omul-din-mijloc fără să mai fie necesar să facă efortul de a intercepta vreo linie telefonică. În schimb, ea trebuie să pătrundă într-un server DNS şi să schimbe o înregistrare, ceea ce este mult mai uşor.

Cum ar putea Trudy să înşele DNS-ul? Acest lucru se dovedeşte a fi relativ uşor. Pe scurt, Trudy poate să păcălească serverul DNS de la ISP-ul lui Alice, făcându-l să trimită o cerere pentru a afla adresa lui Bob. Din nefericire, deoarece DNS-ul foloseşte UDP, serverul DNS practic nu poate verifica cine a dat cu adevărat

răspunsul. Trudy poate profita de această proprietate pentru a falsifica răspunsul aşteptat, introducând astfel o adresă IP falsă în memoria ascunsă a serverului DNS.

Pentru simplitate, vom presupune că iniţial ISP-ul lui Alice nu are o intrare pentru situl Web al lui Bob, *bob.com.* Dacă are, Trudy poate să aştepte până când aceasta expiră şi să încerce mai târziu (sau să folosească alte manevre).

Trudy începe atacul prin a trimite o cerere de căutare ISP-ului lui Alice, solicitând adresa IP a şi- tului *bob.com.* Cum nu are nici o intrare pentru acest nume, serverul local interoghează serverul de nivel superior pentru domeniul *com* pentru a obţine una. Totuşi, Trudy o ia înaintea serverului *com* şi trimite înapoi un răspuns fals spunând: *„bob.com* este 42.9.9.9" , unde acea adresă IP este a ei. Dacă răspunsul ei fals ajunge primul la ISP-ul lui Alice, va fi memorat în memoria ascunsă şi răspunsul adevărat va fi respins ca răspuns nesolicitat la o cerere care nu mai e valabilă. A face un server DNS să instaleze o adresă IP falsă se numeşte **păcălirea DNS-ului** (eng.: **DNS spoofing**). O memorie ascunsă care conţine o adresă IP intenţionat falsă, ca aceasta, se numeşte **memorie ascunsă otăvită (**eng.: **poisoned cache)**.

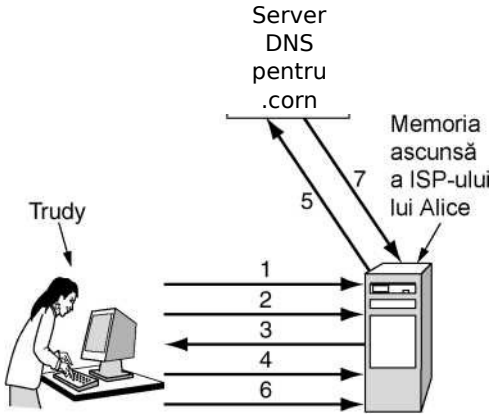
De fapt, lucrurile nu sunt chiar aşa de simple. În primul rând, ISP-ul lui Alice verifică dacă răspunsul are adresa IP sursă corectă a serverului de nivel superior. Dar cum Trudy poate să pună orice doreşte în acel câmp al mesajului, poate trece uşor de acest test, pentru că adresele IP ale serverelor de nivel superior trebuie să fie publice.

În al doilea rând, pentru ca serverele DNS să poată spune care răspuns corespunde cărei cereri, toate cererile poartă un număr de secvenţă. Pentru a păcăli ISP-ul lui Alice, Trudy trebuie să îi cunoască numărul de secvenţă curent. Cea mai uşoară metodă de a afla numărul de secvenţă este ca Trudy să-şi înregistreze ea însăşi un domeniu, să spunem *trudy-the-intruder.com.* Să presupunem că adresa IP a acestuia este tot 42.9.9.9. Ea creează şi un server DNS pentru noul ei domeniu, să spunem *dns.trudy-the-intruder.com.* Şi acesta foloseşte adresa IP a lui Trudy, 42.9.9.9, din moment ce Trudy are un singur calculator. Acum ea trebuie să-l facă pe ISP-ul lui Alice să înregistreze serverul ei DNS. Asta e uşor de făcut. Tot ceea ce are de făcut este să întrebe ISP-ul lui Alice de *foobar.trudy-the-intruder.com*, şi ISP-ul lui Alice va afla cine se ocupă de noul domeniu al lui Trudy întrebând serverul *com* de nivel superior.

Cu *dns.trudy-the-intruder.com* aflat în siguranţă în memoria ascunsă a ISP-ului lui Alice, adevăratul atac poate începe. Acum Trudy întreabă ISP-ul lui Alice de *www.trudy-the-intruder.com.* Bineînţeles, ISP-ul trimite serverului DNS al lui Trudy o cerere referitoare la acest domeniu. Această cerere poartă numărul de secvenţă pe care îl caută Trudy. Sprintenă ca un iepuraş, Trudy îi cere ISP-ului lui Alice să îl caute pe Bob. Apoi răspunde imediat la propria întrebare trimiţând ISP-ului un răspuns falsificat, ca din partea serverului *com* de nivel superior, spunând: *„bob.com* este 42.9.9.9 ". Acest răspuns falsificat poartă un număr de secvenţă cu 1 mai mare decât cel pe care tocmai l-a primit ea. Dacă tot a ajuns aici, poate să mai trimită şi un al doilea fals cu un număr de secvenţă cu 2 mai mare şi poate încă o duzină de răspunsuri false cu numere de secvenţă consecutive. Unul dintre ele va trebui să se potrivească. Restul pur şi simplu vor fi respinse. Când răspunsul fals ajunge la Alice, este stocat în memoria ascunsă; mai târziu, când va ajunge răspunsul adevărat,

va fi respins pentru că nu va mai exista pentru el o cerere valabilă.

Acum, când Alice caută *bob.com,* i se spune să folosească 42.9.9.9, adresa lui Trudy. Trudy a organizat cu succes un atac omul-din-mijloc din confortabila ei cameră de zi. Diverşii paşi ai acestui atac sunt ilustraţi în fig. 8-47. Pentru ca situaţia să fie şi mai complicată, acesta nu e singurul mod de a păcăli DNS-ul. Mai există multe alte moduri.



1. Caută foobar.trudy-the-intruder.com (pentru a-i forţa intrarea în memoria ascunsă a ISP-ului)
2. Caută www.trudy-the-intruder.com (pentru a afla următorul număr de secvenţă al ISP-ului)
3. Cerewww. Trudy-the-intruder.com (Purtând următorul număr de secvenţă al ISP-ului, n)
4. Sprintenă ca un iepuraş, caută bob.com (pentru a forţa ISP-ul să interogheze serverul corn în pasul 5)
5. Legitimează cererea pentru bob.com cu numărul de secvenţă = n+1
6. Răspunsul fals al lui Trudy: Bob este 42.9.9.9, numărul de secvenţă = n+1
7. Răspunsul real (prea târziu)

Fig. 8-47. Cum păcăleşte Trudy ISP-ul lui Alice

DNS sigur

Acest atac, în particular, poate fi împiedicat făcând serverele DNS să folosească identificatori aleatorii în cererile lor în loc să numere pur şi simplu, dar se pare că de fiecare dată când o gaură e astupată, se iveşte una nouă. Adevărata problemă este că DNS-ul a fost proiectat într-o vreme în care Internetul era o facilitate pentru cercetare pentru câteva sute de universităţi şi nici Alice, nici Bob, nici Trudy nu intraseră încă în joc. Pe atunci securitatea nu era o problemă; problema era de a face Internetul să funcţioneze. Situaţia s-a schimbat radical cu trecerea anilor, aşa că în 1994 IETF a înfiinţat un grup de lucru care să facă DNS-ul fundamental sigur. Acest proiect este cunoscut sub numele de **DNSsec** (**DNS security,** rom.: securitatea DNS-ului); Rezultatele lui sunt prezentate în RFC 2535. Din nefericire, DNSsec nu a fost încă distribuit peste tot, aşa că multe servere DNS sunt încă vulnerabile la atacuri de păcălire.

Conceptual, DNSsec este extrem de simplu. Este bazat pe criptografia cu chei publice. Fiecare zonă DNS (în sensul din figura7-4) are o pereche de chei formată dintr-una publică şi una privată. Toate informaţiile trimise de un server DNS sunt semnate cu cheia privată a zonei de origine, deci destinatarul poate să le verifice autenticitatea.

DNSsec oferă trei servicii fundamentale:

1. Dovada originii datelor.
2. Distribuţia cheilor publice.
3. Autentificarea tranzacţiilor şi a cererilor.

Principalul serviciu este primul dintre ele, care verifică faptul că datele returnate au fost aprobate de către deţinătorul zonei. Al doilea este util pentru stocarea şi refacerea, în siguranţă, a cheilor publice. Al treilea este necesar pentru a proteja împotriva atacurilor prin reluare şi a celor de păcălire. Observaţi că păstrarea secretului nu face parte dintre serviciile oferite deoarece toate informaţiile din DNS sunt considerate publice. Cum etapele dezvoltării DNSsec se estimează că vor dura mai mulţi ani, capacitatea serverelor ce pot asigura securitatea de a colabora cu cele ce nu o asigură este esenţială, ceea ce implică faptul că protocolul nu poate fi schimbat. Să vedem acum nişte detalii.

Înregistrările DNS sunt grupate în mulţimi numite **RRSets** (**Resource Record Sets**, rom.: Mulţimi de Înregistrări de Resurse), toate înregistrările cu acelaşi nume, clasă şi tip fiind incluse într-o mulţime. O mulţime poate conţine mai multe înregistrări A, de exemplu, dacă un nume DNS corespunde unei adrese IP primare şi unei adrese IP secundare. Mulţimile sunt extinse cu câteva tipuri noi de înregistrări (discutate mai jos). Pentru fiecare mulţime se face un cod de dispersie criptografic (de exemplu utilizând MD5 sau SHA-1). Rezumatul e semnat de către zona privată a RRSet. La primirea unei mulţimi semnate, clientul poate verifica dacă a fost semnată cu cheia privată a zonei de origine. Dacă semnătura coincide, datele sunt acceptate. Cum fiecare mulţime are propria semnătură, RRSet poate fi memorat oriunde, chiar pe un server neautorizat, fără să pună în pericol securitatea.

DNSsec introduce câteva tipuri noi de înregistrări. Primul din ele este înregistrarea *KEY.* Această înregistrare conţine cheia publică a unei zone, utilizatorul, gazda, sau alt principal, algoritmul criptografic folosit pentru semnături, protocolul folosit pentru transmisie şi încă biţi în plus. Cheia publică este stocată ca atare. Certificatele X.509 nu sunt utilizate, din cauza dimensiunii lor. Câmpul corespunzător algoritmului este 1 pentru semnături MD5/RSA (varianta preferată), sau are alte valori pentru alte combinaţii. Câmpul corespunzător protocolului poate indica folosirea IPsec sau a altor protocoale de securitate, dacă s-a utilizat vreunul.

Al doilea tip nou de înregistrare este *SIG.* Acesta conţine codul de dispersie semnat conform algoritmului specificat în înregistrarea *KEY.* Semnătura se aplică tuturor resurselor din set, inclusiv celor de tip *KEY,* dar nu şi lui însuşi. Înregistrarea mai conţine şi momentele când începe perioada de valabilitate a semnăturii şi când aceasta expiră, precum şi numele celui care semnează şi alte câteva informaţii.

DNSsec este proiectat astfel încât cheia privată a unei zone să poată fi păstrată pe un calculator neconectat la reţea. O dată sau de două ori pe zi, conţinutul bazei de date a unei zone poate fi transportat manual (de exemplu, pe CD-ROM) la o maşină neconectată unde se găseşte cheia privată. Toate mulţimile de înregistrări pot fi semnate acolo şi înregistrările SIG create astfel pot fi duse înapoi, pe CD-ROM, la serverul primar al zonei. Astfel, cheia privată poate fi memorată pe un CD-ROM încuiat într-un seif, mai puţin în momentele în care este introdus în calculatorul neconectat pentru a semna înregistrările noi. Când semnarea s-a terminat, toate copiile cheii sunt şterse din memorie şi CD-ROM-ul este pus înapoi în seif. Această procedură reduce securitatea electronică la securitatea fizică, cu care oamenii ştiu cum să se descurce.

Această metodă de a pre-semna mulţimile de înregistrări accelerează foarte mult procesul de răspuns la cereri, deoarece nu mai trebuie făcute pe loc operaţii de criptografie. Dezavantajul este că e necesară o cantitate mare de spaţiu pe disc pentru a stoca toate cheile şi semnăturile în bazele de date DNS. Unele înregistrări îşi vor mări dimensiunea de zece ori din cauza semnăturii.

Când un proces client obţine o mulţime de înregistrări semnată, trebuie să aplice cheia publică a zonei de origine pentru a decripta rezumatul, să calculeze el însuşi rezumatul, şi să compare cele două valori. Dacă sunt identice, datele sunt

considerate valide. Totuşi, această procedură ridică întrebarea cum va afla clientul cheia publică a zonei. Un mod este de a o cere unui server autorizat, folosind o conexiune sigură (de ex., folosind IPsec).

Totuşi, în practică, e de aşteptat ca clienţii să fie reconfiguraţi cu cheile publice ale tuturor domeniilor de nivel superior. Dacă acum Alice vrea să viziteze situl Web al lui Bob, poate cere DNS- ului mulţimea de înregistrări pentru *bob.com,* care va conţine adresa IP a acestuia şi o înregistrare de tip *KEY* cu cheia publică a lui Bob. Această mulţime de înregistrări va fi semnată de către nivelul superior *com*, deci Alice îi poate verifica uşor validitatea. Un exemplu de ce ar putea conţine această mulţime de înregistrări se află în fig. 8-48.

Numele domeniului	Timp de viaţă	Clasă	Tip	Valoare
bob.com	86400	IN	A	36.1.2.3
bob.com	86400	IN	KEY	3682793A7B73F731029CE2737D
bob.com	86400	IN	SIG	86947503A8B848F5272 E53939C

Fig. 8-48. Un exemplu de mulţime de înregistrări (RRSet) pentru *bob.com.* Înregistrarea *KEY*este cheia publică a lui Bob. Înregistrarea *SIG* este rezumatul serverului *com* de nivel superior pentru înregistrările A şi *KEY* , pentru a le verifica autenticitatea.

Înarmată acum cu o copie verificată a cheii publice a lui Bob, Alice poate întreba serverul DNS al lui Bob (rulat de Bob) de adresa IP pentru *www.bob.com*. Această mulţime de înregistrări va fi semnată cu cheia privată a lui Bob, deci Alice poate verifica semn ătura mulţimii de înregistrări pe care o returnează Bob. Dacă Trudy reuşeşte în vreun fel să introducă o mulţime de înregistrări falsă în una din memoriile ascunse, Alice poate detecta cu uşurinţă lipsa ei de autenticitate pentru că înregistrarea *SIG* va fi incorectă.

Totuşi, DNSsec oferă şi un mecanism criptografic de a lega un răspuns de o anumită cerere, pentru a preveni înşelătoria pe care Trudy a reuşit să o facă în fig. 8-47. Această măsură (opţională) de prevenire a înşelătoriilor adaugă la răspuns un cod de dispersie al mesajului de interogare semnat cu cheia privată a celui care răspunde. Cum Trudy nu ştie cheia privată a serverului *com* de nivel superior, nu poate falsifica un răspuns la o cerere pe care ISP-ul lui Alice a trimis-o acolo. Desigur, ea poate să trimită prima răspunsul, dar acesta va fi respins din cauza semnăturii invalide a rezumatului cererii.

DNSsec suportă şi alte câteva tipuri de înregistrări. De exemplu, înregistrarea *CERT* poate fi utilizată pentru a stoca certificate (de ex., certificate X.509). Această înregistrare a fost oferită pentru că unele persoane vor să transforme DNS-ul într-un PKI. Dacă aceasta se va întâmpla cu adevărat rămâne de văzut. Vom pune capăt aici discuţiei despre DNSsec. Pentru mai multe detalii, vă rugăm să consultaţi RFC 2535.

Nume cu auto-certificare

DNS sigur nu este singura posibilitate de a proteja numele. O abordare complet diferită este întâlnită în **Sistemul Sigur de Fişiere (Secure File System)** (Mazieres

et. al, 1999). În acest proiect, autorii au conceput un sistem de fişiere sigur, scalabil, de întindere foarte largă, fără a modifica DNS-ul (standard) şi fără a folosi certificate sau a presupune existenţa unei PKI. În această secţiune vom arăta cum au putut fi aplicate ideile lor în cazul Web-ului. Prin urmare, în descrierea de mai jos vom folosi terminologia Web în loc de cea a sistemelor de fişiere, utilizată în lucrarea respectivă. Dar, pentru a evita orice confuzie, deşi această schemă *ar putea* să fie aplicată Web-ului pentru a obţine o securitate înaltă, ea nu este folosită în prezent şi ar avea nevoie de modificări substanţiale ale software-ului pentru a fi introdusă.

Vom începe prin a presupune că fiecare server Web are o pereche de chei formată dintr-o cheie publică şi una privată. Esenţa ideii este că fiecare URL conţine un cod de dispersie criptografic pentru numele serverului şi cheia publică a acestuia, ca parte a URL-ului. De exemplu, în fig. 8-49 vedem URL-ul pentru fotografia lui Bob. Începe cu obişnuita schemă *http* , urmată de numele DNS al serverului *(www.bob.com).* Apoi urmează două puncte şi un rezumat de 32 de caractere. La sfârşit este numele fişierului, din nou în modul obişnuit. Cu excepţia codului de dispersie, acesta este un URL standard. Împreună cu codul, este un **URL cu auto-certificare (**eng.: **self-certifymg URL)**.

I Server SHA-l (Server, Cheia publică a serverului) Numele fişierului

http://www.bob.com:2g5hd8bfjkc7mf6hg8dgany23xds4pe6/photos/bob.jpg

Fig. 8-49. Un URL cu auto-certificare conţinând un rezumat al numelui serverului şi al cheii publice.

Întrebarea evidentă este: La ce foloseşte codul de dispersie? Codul e calculat concatenând numele DNS al serverului cu cheia publică a acestuia şi aplicând rezultatului funcţia SHA-l pentru a obţine un cod pe 160 de biţi. În această schemă, rezumatul e reprezentat ca o secvenţă de 32 de cifre şi litere mici, cu excepţia literelor „l" şi „o" şi a cifrelor „1" şi „0", pentru a evita confuziile. Deci au mai rămas 32 de cifre şi de litere posibile. Fiecare dintre cele 32 de caractere disponibile poate codifica un şir de 5 biţi. Un şir de 32 de caractere poate reprezenta rezumatul SHA-l de 160 de biţi. De fapt, nu e necesară folosirea unui rezumat; ar putea fi folosită cheia însăşi. Avantajul rezumatului este că reduce lungimea numelui.

Cel mai simplu (dar cel mai puţin convenabil) mod de a vedea fotografia lui Bob este ca Alice să trimită pur şi simplu şirul din fig. 8-49 programului de navigare. Programul de navigare trimite un mesaj la situl lui Bob, cerându-i cheia publică. Când cheia publică a lui Bob ajunge, programul de navigare concatenează numele serverului şi cheia publică şi execută algoritmul de calcul al rezumatului. Dacă rezultatul coincide cu rezumatul de 32 de caractere din URL-ul securizat, programul de navigare este sigur că are cheia publică a lui Bob. Până la urmă, datorită proprietăţilor algoritmului SHA-l, chiar dacă Trudy interceptează cererea şi falsifică răspunsul, nu are cum să găsească o cheie publică prin care să se obţină rezumatul aşteptat. Orice interferenţă din partea ei va fi astfel detectată. Cheia

publică a lui Bob poate fi stocată în memoria ascunsă pentru utilizări viitoare.

Acum Alice trebuie să verifice că Bob are cheia privată corespunzătoare. Ea construieşte un mesaj conţinând o cheie de sesiune AES propusă, un număr ad-hoc şi o amprentă de timp. Pe urmă criptează mesajul cu cheia publică a lui Bob şi i-l trimite. Cum doar Bob are cheia privată corespunzătoare, numai el poate să decripteze mesajul şi să trimită înapoi numărul ad-hoc criptat cu cheia AES. Când primeşte numărul corect criptat AES, Alice ştie că vorbeşte cu Bob. De asemenea, Alice şi Bob au acum o cheie de sesiune AES pentru următoarele cereri şi răspunsuri *GET.*

Odată ce Alice are fotografia lui Bob (sau orice altă pagină Web), ea poate să o marcheze (eng.: bookmark), ca să nu mai trebuiască să scrie încă o dată întregul URL. Mai mult, URL-urile incluse în paginile Web pot fi de asemenea cu auto-certificare, ca să poată fi folosite doar făcându-se un clic pe ele, dar având în plus şi siguranţa că pagina returnată este cea corectă. Alte moduri de a evita scrierea iniţială a URL-urilor cu auto-certificare sunt obţinerea lor printr-o conexiune sigură de la un server autorizat sau prezenţa lor în certificate X.509 semnate de autorităţi de certificare.

Altă metodă de a obţine URL-uri cu auto-certificare ar fi conectarea la un motor de căutare autorizat, introducând (prima dată) URL-ul cu auto-certificare al acestuia, şi trecerea prin acelaşi protocol cu cel descris mai sus, ceea ce va duce la o conexiune sigură, autentificată, cu motorul de căutare. Apoi motorul de căutare poate fi interogat, iar rezultatele vor apărea pe o pagină semnată, plină cu URL-uri auto-certificate ce pot fi accesate printr-un clic, fără a mai fi necesară scrierea unor şiruri lungi.

Acum să vedem cât de bine rezistă această abordare la păcălelile lui Trudy. Dacă Trudy reuşeşte otrăvirea memoriei ascunse de la ISP-ul lui Alice, cererea lui Alice ar putea fi în mod greşit livrată lui Trudy în loc de Bob. Dar acum protocolul cere ca cel ce primeşte un mesaj iniţial (adică Trudy) să returneze o cheie care să producă rezumatul corect. Dacă Trudy îşi trimite propria cheie publică, Alice o va detecta imediat pentru că rezumatul SHA-l nu se va potrivi cu URL-ul auto-certificat. Dacă Trudy trimite cheia publică a lui Bob, Alice nu va detecta atacul, dar va cripta mesajul următor folosind cheia lui Bob. Trudy va primi mesajul, dar nu va putea să-l decripteze pentru a extrage cheia AES şi numărul ad-hoc. În ambele cazuri, tot ceea ce poate face această păcălire a DNS-ului este să provoace un atac de refuz al serviciului.

8.9.3 SSL – Nivelul soclurilor sigure (Secure Sockets Layer)

Securitatea numelor pe Web reprezintă un bun start, dar există mult mai multe de spus despre securitatea pe Web. Următorul pas îl reprezintă conexiunile sigure. Acum vom discuta despre cum se poate ajunge la conexiuni sigure.

Atunci când Web-ul a intrat în atenţia publicului, el era folosit pentru distribuţia de pagini statice. Oricum, nu după mult timp, câteva companii au avut ideea de a-l folosi pentru tranzacţii financiare, cum ar fi de exemplu cumpărarea de bunuri cu ajutorul cărţii de credit, operaţiuni bancare online şi schimburi electronice de

acţiuni. Aceste aplicaţii au creat o cerere pentru conexiuni sigure. În 1995, compania Netscape Communications, furnizorul dominant de navigatoare de web din acel moment, a răspuns acestei cereri prin introducerea unui pachet de securitate denumit **SSL** ***(Secure Sockets Layer*** rom.: ***nivelul soclurilor sigure****).* Acest program împreună cu protocolul său sunt larg folosite acum şi de Internet Explorer, aşa că merită să fie examinat mai în detaliu.

SSL-ul realizează o conexiune sigură între două socluri, precum şi

1. Negocierea parametrilor între client şi server
2. Autentificare mutuală a clientului şi serverului
3. Comunicare secretă
4. Protecţia integrităţii datelor

Am mai văzut aceste elemente şi înainte astfel că nu este nevoie să insistăm asupra lor în continuare.

Poziţia SSL-ului în cadrul stivei de protocoale este ilustrată în fig. 8-50. De fapt, este un nou nivel interpus între nivelul aplicaţie şi cel de transport, acceptând cereri din partea unui program de navigare şi trimiţându-le mai jos către TCP pentru transmisia către server. Din momentul în care s-a realizat conexiunea sigură, principala sarcină a SSL-ului constă în asigurarea compresiei şi a criptării. Când HTTP este folosit deasupra SSL-ului, este denumit **HTTPS** (rom.: *HTTP securizat – eng.: Secure HTTP),* deşi este vorba de protocolul standard HTTP. Câteodată este disponibil la un nou port (443) în loc de portul standard (80). Ca o particularitate, folosirea SSL nu este restricţionată numai cu programe de navigare Web, deşi aceasta este aplicaţia cea mai des întâlnită.

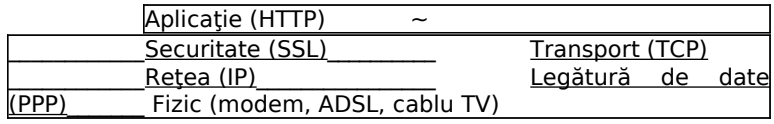


Fig. 8-50. Niveluri (şi protocoale) pentru un utilizator obişnuit ce navighează cu SSL.

Protocolul SSL a cunoscut mai multe versiuni. În cele ce urmează, vom discuta numai despre versiunea 3, care este cea mai folosită. SSL suportă o diversitate de algoritmi şi opţiuni diferite. Aceste opţiuni includ prezenţa sau absenţa compresiei, algoritmii de criptare ce vor fi folosiţi şi câteva chestiuni legate de restricţiile de export impuse criptografiei. Cea din urmă este gândită în principal pentru a asigura că criptografia serioasă este folosită numai în cazul în care ambele capete ale conexiunii sunt situate în Statele Unite. În alte cazuri, cheile sunt limitate la 40 de biţi, ceea ce pentru criptografi reprezintă o glumă. Netscape a fost forţat să impună această restricţie pentru a putea obţine o licenţă de export din partea Guvernului Statelor Unite.

SSL constă din două subprotocoale, unul pentru stabilirea unei conexiuni sigure şi unul pentru folosirea acesteia. Să începem mai întâi să vedem cum sunt stabilite conexiunile securizate. Subprotocolul de stabilire a conexiunii este arătat în fig. 8-51. El începe cu mesajul 1 când Alice trimite o cerere către Bob pentru stabilirea unei legături. Cererea specifică versiunea de SSL pe care o are Alice şi preferinţele

sale cu privire la algoritmii de compresie şi de criptare. De asemenea, conţine un număr ad-hoc (eng.: nonce), R_A, ce va folosit mai târziu.

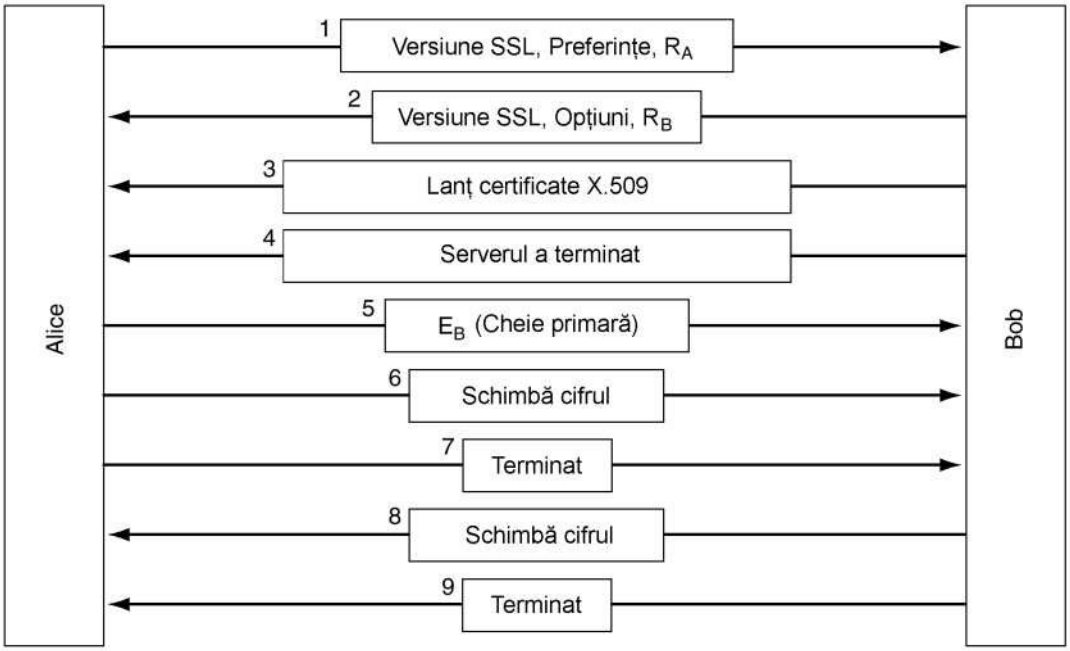


Fig. 8-51. O versiune simplificată a subprotocolului de stabilire unei conexiuni SSL

Acum este rândul lui Bob. În cel de al doilea mesaj, Bob face o alegere între diverşii algoritmi pe care Alice îi poate suporta şi îşi trimite numărul lui ad-hoc, R_B. Apoi, în mesajul 3, el trimite un certificat ce conţine cheia lui publică. Dacă acest certificat nu este semnat de o autoritate recunoscută, el trimite de asemenea şi un lanţ de certificate ce pot fi urmate în sens invers până la ultimul. Toate programele de navigare, inclusiv cel al lui Alice, au predefinite aproximativ 100 de chei publice, aşa că dacă Bob poate stabili un lanţ care ajunge la una din acestea, Alice va putea să verifice cheia publică a lui Bob. În acest moment, Bob poate trimite şi alte mesaje (cum ar fi o cerere pentru certificatul cheii publice al lui Alice). Când Bob a terminat, el trimite mesajul numărul 4 pentru a-i spune lui Alice că este rândul ei.

Alice răspunde prin alegerea unei **chei primare** (eng.: *premaster key)* aleatoare de 384 de biţi şi prin trimiterea acesteia lui Bob, criptată cu cheia lui publică (mesajul numărul 5). Cheia sesiunii folosită de fapt pentru criptarea datelor este derivată din cheia primară combinată cu ambele numere ad-hoc într-o manieră complexă. După ce mesajul 5 a ajuns, atât Alice cât şi Bob sunt în stare să calculeze cheia sesiunii. Datorită acestui fapt, Alice îi spune lui Bob să folosească noul cifru (mesajul 6) şi de asemenea că a terminat subprotocolul de stabilire (mesajul 7). Bob îi răspunde afirmativ la aceste mesaje (mesajele cu numerele 8 şi 9).

Totuşi, deşi Alice ştie cine este Bob, Bob nu ştie cine este Alice (în afara cazului în care Alice are o cheie publică şi un certificat corespunzător pentru ea, o situaţie puţin probabilă în cazul unui individ). De aceea, primul mesaj al lui Bob poate foarte

bine să fie o cerere pentru ca Alice să se autentifice folosind un nume de login şi o parolă ce au fost stabilite anterior. Protocolul de autentificare este în afara domeniului de acoperire al SSL-ului. De îndată ce a fost îndeplinit, prin orice mijloace, transportul de date poate începe.

Aşa cum este menţionat şi mai sus, SSL suportă mai mulţi algoritmi de criptare. Cel mai puternic dintre aceştia foloseşte triplu DES cu trei chei separate pentru criptare şi SHA-l pentru integritatea mesajului. Această combinaţie este relativ lentă, aşa că este folosită de cele mai multe ori de operaţiile bancare şi alte aplicaţii în care este necesar cel mai mare nivel de securitate. Pentru aplicaţii obişnuite de comerţ electronic este folosit RC4 cu o cheie de 128 de biţi pentru criptare şi MD5 pentru autentificarea mesajului. RC4 foloseşte cheia de 128 de biţi ca un punct de plecare şi o extinde la un număr mult mai mare pentru uzul intern. Apoi foloseşte acest număr intern pentru a genera un şir- cheie. Acesta este supus unei operaţii SAU EXCLUSIV cu textul pentru a genera un flux de cifru clasic, aşa cum am văzut în fig. 8-14. Versiunile de export folosesc de asemenea RC4 cu chei de 128 de biţi, dar 88 de biţi dintre aceştia sunt făcuţi public pentru a face cifrul mai uşor de spart.

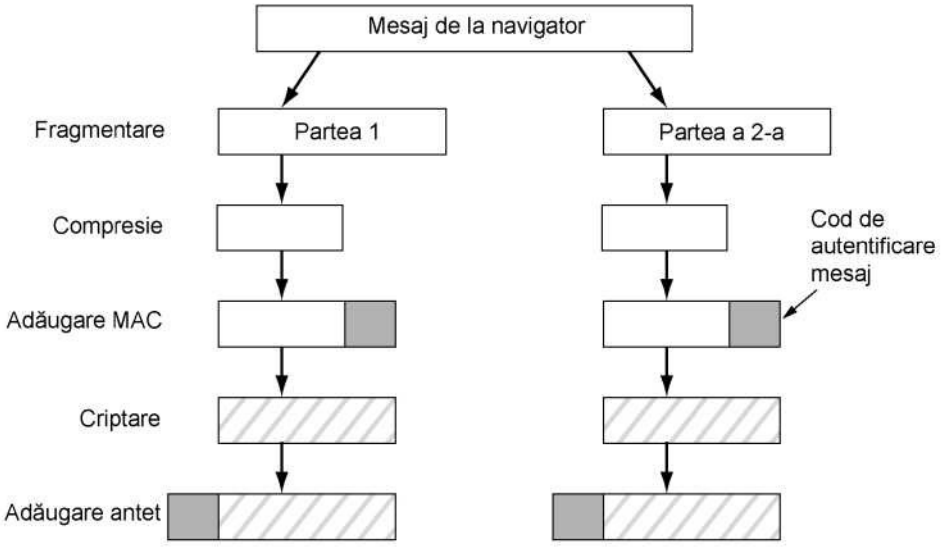


Fig. 8-52. Transmisie de date folosind SSL

Pentru transportul efectiv, este folosit un al doilea subprotocol, aşa cum este arătat în fig. 8-52. Mesajele de la navigator sunt mai întâi sparte în bucăţi de până la 16 KB. Dacă compresia este activată, atunci fiecare unitate este apoi compresată. După aceea, o cheie secretă derivată din cele două numere ad-hoc şi cheia primară este concatenată cu textul compresat şi rezultatul este rezumat cu algoritmul de dispersie agreat (de obicei MD5). Acest cod de dispersie este adăugat fiecărui fragment ca MAC *(Message authentication code – cod pentru autentificarea mesajului).* Fragmentul compresat împreună cu MAC este apoi criptat cu algoritmul de criptare simetrică agreat (de obicei prin aplicarea operaţiei SAU EXCLUSIV cu şirul-cheie RC4). În final, este ataşat un antet de fragment şi fragmentul este transmis prin intermediul conexiunii TCP.

Prudenţa este totuşi importantă. Deoarece a fost demonstrat că RC4 are câteva chei slabe ce pot fi analizate criptologic, securitatea SSL folosind RC4 este pe un teren şubred (Fluhrer et. al, 2001). Programele de navigare ce permit utilizatorului să aleagă suita de cifruri ar trebui configurate să folosească mereu DES triplu cu chei de 168 de biţi şi SHA-l, chiar dacă această combinaţie este mai lentă decât RC4 şi MD5.

O altă problemă cu SSL o reprezintă faptul că principalii pot să nu aibă certificate şi chiar dacă au, ei nu verifică întotdeauna dacă cheile folosite se potrivesc cu certificatele.

În 1996, compania Netscape Communications a trimis SSL către IETF pentru standardizare. Rezultatul a fost **TLS** (eng.: *Transport Layer Security* – *rom.: Securitate la nivelul de transport).* Acesta este descris în RFC 2246.

Schimbările aduse SSL au fost relativ mici, dar suficiente pentru ca versiunea 3 de SSL şi TLS să nu poată coopera. De exemplu, modul în care cheia de sesiune este derivată din cheia primară (eng.: premaster key) şi numerele ad-hoc a fost schimbat pentru a face cheia mult mai puternică (adică mai greu de criptanalizat). Versiunea de TLS este cunoscută ca versiunea 3.1 SSL. Prima implementare a apărut în anul 1999, dar încă nu este clar dacă TLS va înlocui SSL în practică, deşi este puţin mai puternic. Problema cu cheile slabe RC4 rămâne totuşi.

8.9.4 Securitatea codului mobil

Denumirile şi conexiunile sunt două zone de interes pentru securitatea pe Web. Dar sunt mai multe. La început, când paginile Web erau doar fişiere statice HTML, ele nu conţineau cod executabil. Acum, ele conţin adesea mici programe, inclusiv applet-uri Java, programe ActiveX şi fragmente JavaScript. Descărcarea şi execuţia de **cod mobil** (eng.: *mobile code)* prezintă un risc de securitate mare; de aceea au fost elaborate diverse metode pentru micşorarea acestui risc. Vom analiza acum pe scurt unele probleme ce privesc codul mobil şi câteva dintre soluţiile de rezolvare.

Securitatea applet-urilor Java

Applet-urile Java sunt mici programe Java care se traduc în limbajul unei maşini cu stivă denumit **JVM** (eng.: *Java Virtual Machine* – rom.: *Maşină virtuală Java).* Ele pot fi puse intr-o pagină Web fiind descărcate împreună cu pagina. După ce pagina a fost descărcată, applet-urile sunt date unui interpretor JVM din programul de navigare, aşa cum este ilustrat în fig. 8-53.

Avantajul execuţiei codului interpretat faţă de codul compilat este că fiecare instrucţiune este examinată de către interpretor înaintea execuţiei. Astfel este dată o şansă interpretorului de a verifica dacă adresa instrucţiunii este validă. În plus, apelurile de sistem sunt, de asemenea, captate şi interpretate. Felul în care aceste apeluri sunt manipulate este conform politicii locale de securitate. De exemplu, dacă un applet este de încredere (de exemplu, provine de pe discul local), apelurile de sistem pot fi executate fără probleme. Totuşi, dacă un applet este nesigur (de exemplu, a venit din Internet), el poate fi încapsulat în ceea ce se numeşte un **mediu protejat** (eng.: *sandbox)* pentru a restricţiona comportamentul său şi a controla încercările sale de a folosi resursele sistemului.

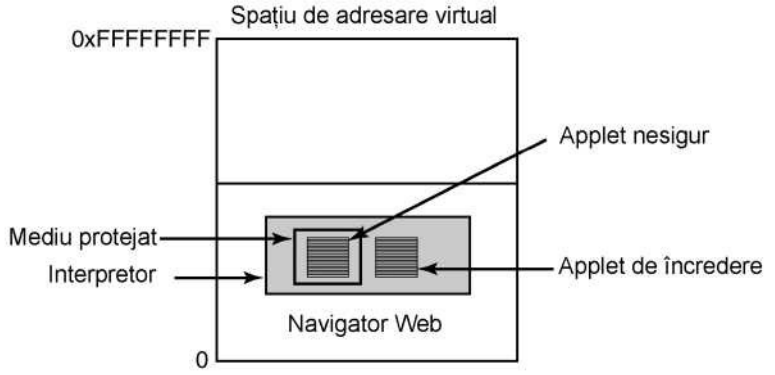


Fig. 8-53. Applet-urile pot fi interpretate de către un navigator Web

Când un applet încearcă să folosesc resursele sistemului, apelul său este trimis pentru aprobare către un monitor de securitate. Acest monitor examinează apelul din punctul de vedere al politici locale de securitate şi apoi decide dacă îl va permite sau îl va respinge. Astfel, este posibil să dea unor applet-uri acces la resurse, dar nu tuturor. Din nefericire, realitatea este că modelul de securitate lucrează prost şi că defectele de programare apar mai tot timpul.

ActiveX

Controalele ActiveX sunt programe binare Pentium ce pot fi incluse în pagini Web. Când este întâlnit unul dintre ele, se efectuează o verificare pentru a vedea dacă ar trebui să fie executat sau nu, şi este executat dacă trece testul. El nu este interpretat şi nici nu este executat într-un mediu protejat, astfel încât codul are aceeaşi putere ca orice alt program şi poate face mult rău. Astfel, întreaga securitate constă în a decide dacă controlul ActiveX va fi sau nu rulat.

Metoda aleasă de Microsoft pentru stabilirea acestei deciziei este bazată pe ideea de **semnare a codului** (eng.: *code signing).* Fiecare program ActiveX vine împreună cu o semnătură digitală – un rezumat al codului ce este semnat de cel ce a creat codul folosind criptografia cu cheie publică. Când un program ActiveX îşi face apariţia, programul de navigare verifică semnătura pentru a se asigura că nu a suferit modificări în timpul transferului. Dacă semnătura este corectă, programul de navigare verifică tabelele sale interne pentru a vedea dacă cel ce a creat programul este de încredere sau există un lanţ ce se termină la un creator de încredere. Dacă creatorul este de încredere, programul este executat; altfel, nu este executat. Sistemul Microsoft care verifică programele ActiveX este numit **Autenticod** (eng.: *Authenticode).*

Este bine să facem o comparaţie între modurile de abordare pentru Java şi ActiveX. Cu Java, nu este important cine a scris applet-ul. În schimb, un interpretor asigură că nu face lucruri interzise de proprietarul maşinii. Prin contrast, cu semnarea codului, nu există nici o încercare de monitorizare a comportamentului codului mobil la execuţie. Dacă el vine de la o sursă de încredere şi nu a fost modificat pe parcurs, el este executat. Nu este făcută nici o încercare de a vedea dacă codul este răuvoitor sau nu. Dacă programatorul original *a* dorit să formateze discul dur şi să şteargă memoria ROM flash, astfel încât calculatorul să nu mai poată fi

pornit, şi dacă programatorul a fost recunoscut ca fiind de încredere, codul va rula şi va distruge calculatorul (excepţie făcând cazul în care controalele ActiveX au fost dezactivate în programul de navigare).

Mulţi oameni simt că încrederea acordată companiilor de programare necunoscute provoacă teamă. Pentru a demonstra problema, un programator din Seattle a format o companie de programare şi a reuşit să o facă recunoscută ca demnă de încredere, ceea ce este foarte uşor de făcut. Apoi a scris un program ActiveX ce realiza oprirea calculatorului şi l-a distribuit. A oprit multe maşini, dar care au fost ulterior re-pornite astfel încât nu au fost pagube. Programatorul a încercat doar să pună în evidenţă problema. Răspunsul oficial a constat în suspendarea certificatului pentru acest program ActiveX, ceea ce a încheiat un scurt episod destul de jenant, dar problema nu a dispărut şi poate fi încă exploatată de către un programator diabolic (Garfinkel cu Spafford, 2002). Deoarece nu există nici o cale de a supraveghea mii de companii de programare ce ar putea să scrie cod mobil, tehnica semnării codului este un dezastru ce aşteaptă să se producă.

JavaScript

JavaScript nu are nici un model formal de securitate, dar are o lungă istorie de implementări vulnerabile. Fiecare furnizor tratează problema securităţii într-o manieră diferită. De exemplu, versiunea 2 de Netscape Navigator a folosit ceva apropiat de modelul Java, dar la versiunea 4 a fost abandonată această manieră în favoarea modelului de cod semnat.

Problema fundamentală este că permiţând execuţia de cod străin pe maşina dumneavoastră vă expuneţi multor necazuri. Din punct de vedere al securităţii, este ca şi cum invitaţi un hoţ în casa dumneavoastră şi apoi încercaţi să-l urmăriţi cu atenţie pentru a nu putea ieşi din bucătărie şi intra în sufragerie. Dacă se întâmplă ceva neaşteptat şi sunteţi neatent pentru o clipă, se pot întâmpla lucruri rele. Problema este că codul mobil permite o grafică atractivă şi o interacţiune rapidă şi foarte mulţi proiectanţi Web cred că acestea sunt mult mai importante decât securitatea, în special când este vorba de riscul altcuiva.

Viruşi

Viruşii sunt o altă formă de cod mobil. Spre deosebire de exemplele de mai sus, viruşii nu sunt invitaţi deloc. Diferenţa între un virus şi un cod mobil obişnuit este aceea că viruşii sunt scrişi pentru a se reproduce. Când un virus ajunge într-un calculator, prin intermediul unei pagini Web, al unui document ataşat sau printr-o altă cale, el începe prin infectarea programelor executabile de pe disc. Când unul dintre aceste programe este executat, controlul este transferat virusului, care de obicei încearcă să se întindă pe alte maşini, de exemplu, prin trimiterea de mesaje cu propriile sale copii către toate persoanele din agenda victimei. Unii viruşi infectează sectorul de start al discului dur, astfel că atunci când maşina porneşte, virusul este activat. Viruşii au devenit o problemă uriaşă pe Internet şi au provocat pagube de miliarde de dolari. Nu există o soluţie foarte clară. Probabil că soluţia ar fi cea a unei noi generaţii de sisteme de operare bazate pe micronuclee şi o separare puternică a utilizatorilor, proceselor şi resurselor.

8.10 IMPLICAŢII SOCIALE

Internet-ul şi tehnologiile sale de securitate reprezintă o zonă în care problemele de ordin social, politici publice şi tehnologie se întâlnesc într-o confruntare directă, deseori cu consecinţe grave. În cele ce urmează vom analiza pe scurt trei zone: confidenţialitate, libertatea de exprimare şi dreptul de autor. Nu mai este nevoie să spunem că atingem doar tangenţial aceste subiecte. Pentru lecturi suplimentare, a se vedea (Anderson, 2001; Garfinkel cu Spafford, 2002; Schneier, 2000). Internet-ul este, de asemenea, plin de materiale documentare. Este suficient să folosiţi cuvinte cheie precum

„confidenţialitate" (eng.:*privacy),* „cenzură" (eng.: *censorship)* şi „drept de autor" (eng.: *copyright)* în orice motor de căutare. De asemenea puteţi consulta pagina Web a acestei cărţi pentru alte câteva legături interesante.

8.10.1 Confidenţialitate

Au oamenii dreptul la confidenţialitate? Bună întrebare. Cel de-al patrulea amendament al Constituţiei Statelor Unite împiedică guvernul de-a scotoci casele, documentele şi obiectele personale ale oamenilor fără un motiv întemeiat şi stabileşte circumstanţele în care sunt emise mandatele de percheziţie. Astfel, confidenţialitatea a fost o problemă publică de peste 200 de ani, cel puţin în Statele Unite.

Ceea ce s-a schimbat în ultimul deceniu sunt atât uşurinţa cu care guvernele îşi pot spiona cetăţenii, cât şi uşurinţa cu care cetăţenii pot preveni un astfel de spionaj. În secolul al XVIII-lea, pentru ca guvernul să cerceteze documentele unui cetăţean trebuia să trimită un poliţist călare la ferma cetăţeanului pentru a vedea anumite documente. Era o procedură greoaie. În zilele noastre, companiile de telefonie şi furnizorii de Internet instalează dispozitive de ascultare atunci când le sunt prezentate mandate de percheziţie. Viaţa este mult mai uşoară astfel pentru poliţist şi nici numai există pericolul unei căzături de pe cal.

Criptografia schimbă toate aceste aspecte. Oricine îşi permite deranjul de a descărca şi instala PGP şi foloseşte o cheie bine păzită de o putere extraterestră poate fi destul de sigur că nimeni în universul pe care-l cunoaştem nu îi poate citi mesajele electronicele, cu sau fără mandat de percheziţie. Guvernele înţeleg foarte bine această chestiune şi nu o agreează. Confidenţialitate reală înseamnă dificultăţi mai mari în a spiona criminalii de orice fel, dar şi greutăţi în a spiona jurnalişti şi opo- nenţi politici. Prin urmare, unele guverne restricţionează sau interzic folosirea sau exportul de criptografie. În Franţa, de exemplu, până în 1999, toată criptografia era interzisă dacă cheile nu erau date de către guvern.

Franţa nu a fost o excepţie. În aprilie 1993, guvernul Statelor Unite a anunţat intenţia sa de realiza un criptoprocesor hardware, **cip-ul clipper** (eng.: *clipper chip),* standard pentru toate comunicaţiile în reţea. În acest fel, s-a spus, confidenţialitatea cetăţenilor va fi garantată. De asemenea era menţionat că cip-ul permitea guvernului decriptarea traficului prin intermediul unei scheme numite **custodie de chei** (eng.: *key escrow),* ce permitea guvernului să aibă acces la toate

cheile. Totuşi, se promitea ascultarea mesajelor numai cu un mandat valabil de percheziţie. Nu mai trebuie spus că a urmat o largă dispută, avocaţii confidenţialităţii denunţând tot planul şi avocaţii oficiali lăudându-l. În cele din urmă, guvernul a bătut în retragere şi a renunţat la idee.

O mare cantitate de informaţie despre confidenţialitatea electronică poate fi găsită pe pagina Web a Fundaţiei Graniţei Electronice (eng.: Electronic Frontier Foundation), *www.eff.org*.

Retransmiţătoare anonime

PGP, SSL şi alte tehnologii fac posibil ca două părţi să stabilească o comunicaţie securizată, autentificată, neurmărită de o terţă parte şi fără interferenţe. Totuşi, câteodată, confidenţialitatea este cel mai bine servită de absenţa autentificării, prin realizarea unei comunicaţii anonime. Anonimita- tea poate fi dorită pentru mesaje punct-la-punct, grupuri de ştiri sau în ambele cazuri.

Să considerăm câteva exemple. În primul rând, dizidenţii politici ce trăiesc sub regimuri autoritare doresc deseori să comunice în mod anonim pentru a nu fi trimişi la închisoare sau omorâţi. În al doilea rând, comportamentul ilegal din foarte multe corporaţii, organizaţii educaţionale, guvernamentale şi altele a fost făcut public de către cei interesaţi, care de cele mai multe ori preferă să rămână anonimi pentru a evita plătirea eventualelor poliţe ulterioare. În cel de al treilea rând, oamenii cu vederi sociale, politice sau religioase nepopulare ar dori să comunice unul cu altul prin intermediul mesajelor electronice sau a grupurilor de ştiri fără a se expune. În al patrulea rând, oamenii ar putea dori să discute despre alcoolism, boli mentale, hărţuire sexuală, abuzul asupra copiilor sau despre faptul de a fi membru a unei minorităţi persecutate politic fără a-şi face publice părerile. Există, bineînţeles, numeroase alte exemple.

Să luăm în considerare un exemplu concret. În anii 1990, câţiva critici ai unui grup religios netradiţionalist şi-au publicat părerile pe un grup de ştiri USENET prin intermediul unui **retransmi- ţător anonim** (eng.: *anonymous remailer).* Acel server permitea utilizatorilor să creeze pseudonime şi să trimită mesaje server-ului, care apoi le retransmitea şi le republica folosind pseudonimul, astfel încât nimeni nu şi-a dat seama din partea cui au venit. Unele dintre mesaje arătau ceea ce grupul religios pretindea a fi secrete ale meseriei şi documente cu drept de autor. Grupul religios a răspuns comunicând autorităţilor locale că secretele meseriei sale au fost făcute publice şi că drepturile de autor au fost încălcate, acestea fiind delicte în zona în care server-ul era localizat. A urmat un proces şi operatorul server-ului a fost obligat să predea informaţia care a dezvăluit identităţile persoanelor ce au publicat anunţurile (în mod întâmplător, aceasta nu era prima dată când religia nu era fericită când cineva i-a descoperit secretele: William Tyndale a fost ars pe rug în 1536 pentru că a tradus Biblia în limba engleză).

O destul de mare parte a comunităţii Internet a fost scandalizată de o asemenea încălcare a confidenţialităţii. Concluzia pe care fiecare a tras-o a fost că un retransmiţător anonim care menţine o tabelă de corespondenţe între adrese de poştă reale şi pseudonime (numit retransmiţător de tipul 1) nu valorează prea mult. Acest caz a stimulat diverse persoane să conceapă retransmiţătoare anonime ce ar putea rezista atacurilor cu citaţie (eng.: *subpoena attacks*).

Aceste retransmiţătoare noi, deseori denumite **retransmiţătoare de tip**

cypherpunk (eng.: *cypherpunk remailers*), funcţionează după cum urmează. Utilizatorul produce un mesaj electronic, îl completează cu antetele specifice RFC 822 (exclusiv câmpul *De la:* – eng.: *From:),* îl criptează cu cheia publică a retransmiţătorului şi îl trimite retransmiţătorului. Acolo, antetele externe RFC 822 sunt şterse, conţinutul este decriptat şi mesajul este retransmis. Retransmiţătorul nu are conturi şi nici nu păstrează fişiere cu istoria lucrurilor petrecute (eng.: *log-uri*), aşa că, chiar dacă server-ul este mai târziu confiscat, el nu reţine nici o urmă a mesajelor ce au trecut prin el.

Mulţi utilizatori ce doresc să rămână anonimi îşi revendică cererile prin intermediul mai multor retransmiţătoare, aşa cum este ilustrat în fig. 8-54. Astfel, Alice doreşte să-i trimită lui Bob o felicitare de ziua Sfântului Valentin, foarte, foarte, foarte anonimă, astfel că foloseşte trei retransmiţătoare. Ea compune mesajul, *M* şi îi pune un antet ce conţine adresa lui Bob. Apoi îl criptează cu cheia publică a retransmiţătorului 3, E_3 (indicat prin liniile orizontale). La acesta ea prefixează un antet cu adresa retransmiţătorului 3 în text clar. Acest mesaj este arătat între retransmiţătoarele 2 şi 3.

Apoi ea criptează mesajul cu cheia publică a retransmiţătorului 2, E_2 (indicat prin liniile verticale) şi adăugă la început un antet de text clar ce conţine adresa retransmiţătorului 2. Acest mesaj este arătat în fig. 8-54 între 1 şi 2. În cele din urmă, criptează întregul mesaj cu cheia publică a retransmi- ţătorului 1, E_1, şi pune la începutul lui un antet de text clar cu adresa retransmiţătorului 1. acest mesaj este arătat în figură la dreapta lui Alice şi este ceea ce ea transmite de fapt.

Când mesajul ajunge la retransmiţătorul 1, antetul exterior este şters. Corpul mesajului este decriptat şi apoi trimis retransmiţătorului 2. Paşi similari se petrec la nivelul celorlalte două retransmiţătoare.

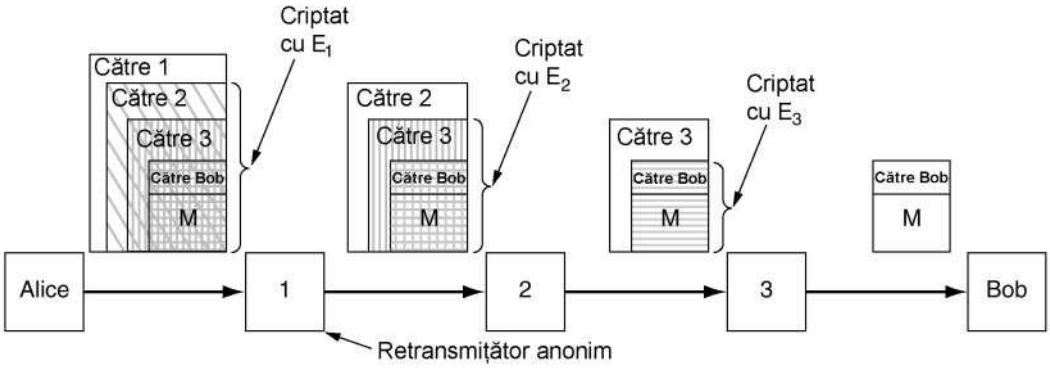


Fig. 8-54. Alice foloseşte 3 retransmiţătoare pentru a trimite lui Bob un mesaj.

Deşi este extrem de dificil pentru cineva să urmărească mesajul final pe calea de retur către Alice, multe retransmiţătoare îşi iau măsuri suplimentare de siguranţă. De exemplu, ele pot păstra mesajele pentru o perioadă aleatoare de timp, pot adăuga sau şterge bucăţi nefolositoare la sfârşitul mesajului, pot reordona mesajele, toate acestea pentru a face dificil oricui să determine ce mesaj de ieşire al unui retransmiţător corespunde unui mesaj de intrare, pentru a îngreuna analiza traficului. Pentru o descriere amănunţită a unui sistem ce reprezintă o capodoperă

în mesageria anonimă, vedeţi (Mazieres şi Kaashoek, 1998).

Anonimitatea nu este restricţionată la mesageria electronică. Există servicii ce permit navigare anonimă pe Web. Utilizatorul îşi configurează programul de navigare pentru a folosi serviciul anonim ca pe un proxy. De aici înainte toate cererile HTTP se vor duce la serviciul anonim, care va cere pagina şi o va trimite înapoi. Pagina Web va vedea că serviciul de „anonim”-izare este sursa a cererii, şi nu utilizatorul. Atâta timp cât serviciul de anonimizare nu va ţine un istoric al activităţii, nimeni nu îşi poate da seama cine a cerut pagina.

8.10.2 Libertatea de exprimare

Confidenţialitatea se referă la indivizii ce doresc să limiteze accesul celorlalţi la informaţii despre propria lor persoană. O a doua problemă socială cheie o reprezintă libertatea de exprimare şi opusa ei, cenzura, care are legătură cu guvernele ce doresc să restricţioneze ceea ce indivizii pot citi şi publica. Deoarece conţine milioane şi milioane de pagini, Web-ul a devenit paradisul cenzurii. Materialul interzis, în funcţie de natura şi ideologia regimului, poate include pagini Web ce conţin elemente din lista următoare:

1. Materiale nepotrivite pentru copii şi adolescenţi.
2. Ură faţă de diverse grupări etnice, religioase, sexuale şi altele.
3. Informaţii despre democraţie şi valori democratice.
4. Relatări ale unor evenimente istorice ce contrazic versiunea guvernamentală.
5. Manuale pentru deschiderea încuietorilor, construirea de arme, criptare de mesaje, etc.

Măsura obişnuită este de a interzice paginile rele.

Câteodată, rezultatele sunt neaşteptate. De exemplu, câteva biblioteci publice au instalat filtre Web pe calculatoarele lor pentru a le face prietenoase faţă de copii, interzicând paginile pornografice. Filtrele înscriu paginile pe listele lor negre, dar în acelaşi timp caută cuvinte nepotrivite în paginile ce urmează a fi afişate. Într-un caz din comitatul Loudoun din Virginia, filtrul a blocat căutarea unor materiale despre cancerul de sân deoarece filtrul a văzut cuvântul „sân”. Patronul bibliotecii a dat în judecată comitatul Loudoun. Pe de altă parte, în Livermore din California, un părinte a dat în judecată biblioteca publică pentru că *nu* a instalat un filtru, după ce băiatul ei de 12 ani a fost prins uitându-se la materiale pornografice. Ce ar trebui să facă o bibliotecă?

A scăpat din vederea multor oameni că World Wide Web este o reţea mondială. Nu toate ţările sunt de acord cu ceea ce ar trebui să fie permis pe Web. De exemplu, în noiembrie 2000, un tribunal francez a dictat corporaţiei californiene Yahoo să blocheze accesul utilizatorilor francezi la licitaţiile de obiecte asociate nazismului de pe pagina Yahoo, deoarece astfel de materiale violează legea franceză. Yahoo a apelat la un tribunal nord-american, care a luat partea corporaţiei, dar problema cu privire la „ce legi se aplică şi unde se aplică” este departe de a fi rezolvată.

Imaginaţi-vă ce s-ar putea întâmpla dacă un tribunal din Utah ar spune Franţei

să blocheze paginile Web despre vin, deoarece nu se conformează cu legile mult mai stricte privitoare la alcool ale statului Utah? Presupuneţi că China ar cere ca toate paginile Web despre democraţie să fie interzise deoarece nu sunt în interesul statului. Oare legile iraniene referitoare la religie se aplică mult mai liberalei Suedii? Poate Arabia Saudită bloca pagini Web ce vorbesc de drepturile femeilor? Toată problema este o veritabilă cutie a Pandorei.

Un comentariu relevant al lui John Gilmore este: „Reţeaua interpretează cenzura ca pe o pagubă şi o ocoleşte". Pentru o implementare concretă, uitaţi-vă la **serviciul etern** (eng.: *eternity service)* (Anderson, 1996). Scopul lui este de a se asigura că o informaţie publicată nu poate fi retrasă sau rescrisă, aşa cum era destul de obişnuit în Uniunea Sovietică în timpul conducerii staliniste. Ca să folosească serviciul eternităţii, utilizatorul trebuie să specifice cât timp trebuie păstrat materialul, să plătească o taxă proporţională cu această durată şi cu volumul materialului şi să-l încarce apoi în Internet. După aceea, nimeni nu-l poate şterge sau edita, nici chiar şi cel care l-a încărcat.

Cum ar putea fi implementat un asemenea serviciu? Modelul cel mai simplu este folosirea unui sistem de la egal la egal (eng.:*peer-to-peer)* în care documentele să fie depuse în câteva zeci de servere, fiecare primind o parte a taxei şi, prin asta, un motiv să se alăture sistemului. Serverele ar trebui împărţite în cât mai multe jurisdicţii legale pentru a obţine o flexibilitate maximă. Liste de câte 10 servere selectate aleator ar fi păstrate în siguranţă în mai multe locuri, astfel încât dacă unele sunt compromise, altele ar continua să furnizeze informaţiile cerute. O autoritate ce doreşte distrugerea documentului nu poate fi niciodată sigură că a găsit toate copiile. De asemenea, sistemul s-ar putea regenera: dacă află că unele copii au fost distruse, siturile rămase ar încerca să găsească noi locuri.

Serviciul eternităţii a fost prima propunere de sistem rezistent la cenzură. De atunci, au fost propuse şi, uneori implementate şi alte sisteme. Au fost adăugate diverse caracteristici noi, cum ar fi criptarea, anonimitatea şi toleranţa la defecte. Deseori, fişierele ce trebuie stocate sunt sparte în mai multe părţi, fiecare memorată pe mai multe servere. Câteva exemple sunt Freenet (Clarke et. al, 2002), PASIS (Wylie et. al, 2000) şi Publius (Waldman et. al, 2000). Alte rezultate sunt prezentate în (Serjantov, 2002).

Tot mai multe ţări încearcă să impună reguli pentru exportul de lucruri imateriale, care includ pagini Web, programe, lucrări ştiinţifice, mesaje electronice, suport telefonic şi altele. Chiar şi Marea Britanie, care are o îndelungată tradiţie a libertăţii de exprimare, se gândeşte la legi mult mai restrictive, care ar defini, de exemplu, discuţiile tehnice dintre un profesor britanic şi studentul său străin la Universitatea din Cambridge ca pe un export legiferat ce are nevoie de o licenţă guvernamentală (Anderson, 2002). Nu este nevoie să mai spunem că asemenea politici sunt controversate.

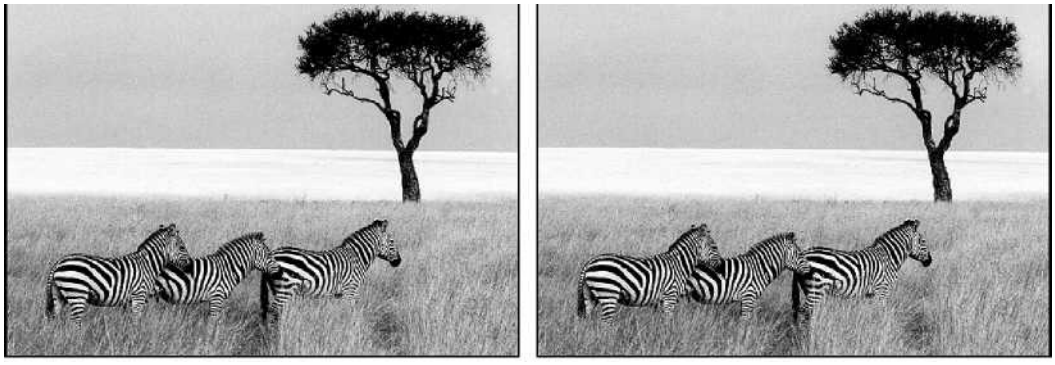
Steganografie

În ţările cu o cenzură puternică, dizidenţii încearcă deseori să folosească tehnologia pentru a eluda cenzura. Criptografia permite trimiterea (nu întotdeauna legală) de mesaje secrete, dar dacă guvernul crede că Alice este o Persoană Rea, simplul fapt că ea comunică cu Bob ar putea să-l pună şi pe el în aceeaşi categorie, deoarece guvernele represive recunosc conceptul de tranzitivitate, chiar dacă nu au

prea mulţi matematicieni. Retransmiţătoarele anonime pot ajuta, dar dacă sunt interzise pe plan intern şi mesajele către retransmiţătoare externe necesită licenţe de export, nu pot ajuta chiar atât de mult. Dar Web-ul poate.

Oamenii care vor să comunice în secret încearcă deseori să ascundă că de fapt comunicare are loc. Ştiinţa ascunderii mesajelor este denumită **steganografie** *(steganography),* din cuvintele greceşti ce s-ar traduce prin „scriere acoperită”. De fapt, grecii antici o foloseau şi ei. Herodot a scris despre un general care a ras în cap un sol, i-a tatuat pe scalp un mesaj şi l-a lăsat să-i crească părul la loc înainte de a-l trimite în misiune. Tehnicile moderne sunt, conceptual, aceleaşi, doar că au o mai mare lărgime de bandă şi o întârziere mai mică.

Ca un caz în speţă, să considerăm fig. 8-55 (a). Această fotografie, realizată în Kenya, conţine trei zebre care contemplă un arbore acacia. În fig. 8-55 (b) se pare că ar fi aceleaşi trei zebre şi arborele acacia, dar există o atracţie specială. Conţine textul complet, necenzurat a cinci piese shakespeariene: *Hamlet, Regele Lear, Macbeth, Negustorul din Veneţia* şi *Iulius Cezar*. Împreună, aceste piese totalizează peste 700 kiloocteţi de text.



(a) (b)

Fig. 8-55. (a) Trei zebre şi un arbore. (b) Trei zebre, un arbore şi textul complet a cinci piese de William Shakespeare.

Cum funcţionează acest canal steganografie? Imaginea color originală este de 1024 x 768 pixeli. Fiecare pixel constă din trei numere de 8 biţi, câte unul pentru fiecare din culorile roşu, verde şi albastru a acelui pixel. Culoarea acelui pixel este formată de o superpoziţie a celor trei culori. Metoda de codificare steganografică foloseşte bitul cel mai puţin semnificativ din fiecare octet al a culorilor RGB ca un canal ascuns. Astfel fiecare pixel are spaţiu pentru 3 biţi de informaţie secretă, unul în valoare roşie, unul în verde şi unul în cea albastră. Într-o imagine de asemenea mărime, se pot memora până la 1024 x 768 x 3 biţi sau 294.912 octeţi de informaţie secretă.

Textul complet al celor cinci piese împreună cu o scurtă notiţă însumează 734.891 octeţi. Acest text a fost mai întâi comprimat la aproximativ 270 KB folosind un algoritm de compresie standard. Rezultatul a fost apoi criptat folosind IDEA şi inserat în biţii cei mai puţin semnificativi ai fiecărui octet de culoare. După cum se poate vedea (de fapt, nu se poate vedea), existenţa informaţiei este total invizibilă.

Este la fel de invizibilă în versiunea mărită, color a imaginii. Ochiul nu poate diferenţia uşor culoarea pe 21 de biţi faţă de cea pe 24 de biţi.

Figurând cele două imagini în alb-negru la o rezoluţie scăzută nu demonstrează cât de puternică este această. Pentru a simţi mai bine cum lucrează steganografia, autorul a pregătit o demonstraţie, cu imaginea color la o rezoluţie mare a fig. 8-55(b) având cele cinci piese incluse în ea. Demonstraţia, inclusiv instrumentele pentru inserarea şi extragerea textelor în şi din imagini, poate fi găsită la pagina Web a cărţii.

Pentru a folosi steganografia pentru comunicaţia nedetectată, dizidenţii ar putea crea o pagină Web plină de imagini politic-corecte, cum ar fi fotografii ale Marelui Lider, sporturi locale, vedete de film şi televiziune, etc. Desigur, imaginile ar fi pline de mesaje steganografice. Dacă mesajele ar fi întâi compresate şi apoi criptate, chiar şi cineva care le-ar suspecta prezenţa ar avea dificultăţi imense în distingerea mesajelor faţă de zgomot. Desigur, imaginile ar trebui să fie scanări recente; copiind o imagine de pe Internet şi schimbând o parte din biţi nu conduc la rezultat.

Imaginile nu sunt singurele purtătoare de mesaje steganografice. Fişierele video sunt de asemenea bune. Fişierele video au o lărgime de bandă steganografică foarte mare. Chiar şi aşezarea şi ordinea etichetelor într-un fişier HTML pot fi purtătoare de informaţie.

Deşi am examinat steganografia în contextul exprimării libere, ea are numeroase alte utilizări. O utilizare banală este codificarea drepturilor de proprietate chiar în imaginea la care se referă aceste drepturi. Dacă o astfel de imagine este furată şi depusă pe o pagină Web, proprietarul de drept poate dezvălui mesajul steganografic în cursul unui proces pentru a proba a cui este imaginea. Această tehnică este cunoscută sub denumirea de **filigranare** (eng.: *watemarking).* Ea este discutată în (Piva et. al, 2002).

Pentru mai multe despre steganografie, vedeţi (Artz, 2001; Johnson şi Jajoda, 1998; Katzenbeisser şi Petitcolas, 2000 şi Wainer. 2002).

8.10.3 Dreptul de autor

Confidenţialitatea şi cenzura sunt doar două din zonele unde tehnologia întâlneşte politicul. O a treia este problema **dreptului de autor** (eng.: *copyright).* Acordarea dreptului de autor înseamnă a garanta creatorului de **PI** *(proprietate intelectuală* – eng.: *Intellectual Propriety),* inclusiv scriitori, artişti, muzicieni, fotografi, cineaşti, coreografi şi alţii, dreptul exclusiv de a exploata proprietatea lor intelectuală pentru o perioadă de timp, de obicei viaţa autorului plus 50 sau 75 de ani în cazul proprietăţii comune. După ce expiră dreptul de autor asupra unei lucrări, aceasta trece în domeniul public şi oricine o poate folosi sau vinde după bunul plac. De exemplu, proiectul Gutenberg *(www.promo.net/pg),* de exemplu, a pus pe Web mii de lucrări din domeniul public (de exemplu, Shakespeare, Twain, Dickens). În anul 1998, Congresul Statelor Unite a extins dreptul de autor la încă 20 de ani la cererea Hollywood-ului, care a motivat cererea prin faptul că fără o extindere a termenului, nimeni nu ar mai crea nimic, niciodată. Prin contrast, patentele durează doar 20 de ani şi totuşi oamenii încă mai inventează lucruri.

Dreptul de autor a ajuns în prim-plan când Napster, un serviciu de traficare a

muzicii, a atins 50 de milioane de membri. Deşi Napster nu copia muzica, tribunalele au decis că, deoarece deţinea o bază de date ce reflecta cine avea ce cântec, a contribuit la încălcarea legii, cu alte cuvinte a ajutat pe alţii să încălcat legea. Deşi nimeni nu pretinde că dreptul de autor este o idee proastă (mulţi pretind că termenul este mult prea lung, favorizând marile corporaţii faţă de public), noua generaţie de partajare a muzicii are deja implicaţii etice serioase.

De exemplu, să considerăm o reţea de la egal la egal în care oamenii partajează fişiere legitime (muzică din domeniul public, filme video, tratate religioase ce nu sunt secrete înregistrate etc.) şi câteva cu drepturi de autor. Să presupunem că toţi sunt permanent conectaţi prin ADSL sau cablu. Fiecare calculator are un index cu ce este pe discul dur plus o listă a altor membri. Cineva care caută un anumit obiect poate să aleagă aleator un membru şi să vadă dacă obiectul căutat este în posesia lui. Dacă nu, poate verifica toţi membrii din lista acelei persoane şi toţi membri din listele lor ş.a.m.d. Calculatoarele sunt foarte bune la astfel de lucruri. Dacă a găsit elementul respectiv, solicitantul doar îl copiază.

Dacă acel obiect este cu drept de autor, există şanse ca solicitantul să încalce legea (deşi pentru transferuri internaţionale, nu este clar ce legi sunt aplicabile). Dar ce putem spune despre furnizor? Este un delict să păstrezi muzică pentru care ai plătit şi ai descărcat-o legal pe discul tău dur unde alţii o pot găsi? Dacă aveţi o cabană neîncuiată la ţară şi un hoţ de PI se furişează purtând un calculator portabil şi un scanner, copiază o carte ce are drepturi de autor şi apoi pleacă, eşti *tu* vinovat de delictul de a nu fi protejat dreptul de autor al altcuiva?

Dar este mai mult de discutat pe tema dreptului de autor. Există o mare dispută între Hollywood şi industria calculatoarelor. Primul doreşte protecţie la sânge a întregii proprietăţi intelectuale şi cel de-al doilea nu vrea să fie poliţistul Hollywood-ului. În octombrie 1998, congresul a votat **Legea dreptului de autor digital** (eng.: *Digital Millennium Copyright Act – DMCA)* care stipulează că este delict să treci peste orice mecanism de protecţie a unei lucrări cu drept de autor sau să spui altora cum să o facă. O legislaţie similară este pusă în practică şi în Uniunea Europeană. În timp ce nimeni nu crede că piraţilor din extremul Orient ar trebui să le fie permisă copierea lucrărilor cu drept de autor, mulţi oameni cred că DMCA schimbă total balanţa între interesul proprietarului de drept de autor şi interesul public.

Un exemplu! În septembrie 2000, un consorţiu al industriei muzicale însărcinat să realizeze un sistem infailibil pentru a vinde de muzică online a sponsorizat un concurs, invitând oamenii să încerce să spargă sistemul (care este chiar lucrul cel mai potrivit pe care să-l faci cu un sistem de securitate nou). O echipă de cercetători în securitate de la diverse universităţi, condusă de profesorul Edward Felten din Princeton, a primit provocarea şi a spart sistemul. Apoi au scris o lucrare despre ce au găsit ei şi au trimis-o unei conferinţe USENIX de securitate, unde a fost recenzată şi acceptată. Înainte de prezentarea lucrării, Felten a primit o scrisoare de la Asociaţia Industriei Americane de Înregistrări (eng.: *Recording Industry Association of America)* ce ameninţa autorii cu un proces ce avea la bază DMCA dacă ar fi publicat lucrarea.

Răspunsul lor a fost intentarea unui proces, rugând un tribunal federal să decidă dacă este încă legal să publici lucrări ştiinţifice despre securitate. Cu spaima unei

decizii definitive împotriva lor, industria şi-a retras plângerea şi tribunalul a casat procesul lui Felten. Fără îndoială că industria a fost motivată de slăbiciunea cauzei lor: ei au invitat oamenii să încerce să le spargă sistemul şi apoi tot ei au ameninţat cu darea în judecată pe câţiva care au acceptat provocarea. Cu ameninţarea retrasă, lucrarea a fost publicată (Craver et. al, 2001). O nouă confruntare este inevitabilă.

O problemă apropiată este răspândirea doctrinei de utilizare corectă (eng.: *fair use doctrine),* ce a fost stabilită prin hotărâri judecătoreşti în diverse ţări. Această doctrină spune că cumpărătorii lucrării cu drept de autor au câteva drepturi limitate de a copia lucrarea, inclusiv dreptul de a cita părţi din ea pentru scopuri ştiinţifice, de a o folosi în scopuri didactice în şcoli şi licee şi în unele cazuri de a face copii de rezervă pentru folosul personal, în cazul în care mediul original nu mai este bun. Testele pentru ceea ce constituie o utilizare corectă includ (1) dacă materialul este comercial, (2) ce procentaj din întreg este copiat şi (3) efectul copierii asupra vânzării lucrării. Din moment ce DMCA şi legi similare din Uniunea Europeană întârzie eludarea schemelor de protecţie, aceste legi interzic de asemenea o utilizare corectă. De fapt, DMCA ia utilizatorilor nişte drepturi istorice pentru a da mai multă putere vânzătorilor de conţinut. O confruntare finală este inevitabilă.

O altă evoluţie a lucrurilor ce restrânge chiar şi DMCA-ul în schimbarea balanţei între proprietarii dreptului de autor şi utilizatori este **TCPA** (eng.: *Trusted Computing Platform Alliance* – rom.: *Alianţa platformelor de calcul de încredere)* condusă de Intel şi Microsoft. Ideea este de a face ca microprocesorul şi sistemul de operare să monitorizeze atent comportamentul utilizatorului în diverse moduri (de exemplu, când ascultă muzică pirat) şi să prevină comportamentele nedorite. Sistemul permite chiar deţinătorilor de conţinut să manipuleze la distanţă PC-ul utilizatorului pentru a schimba regulile când aceasta este neapărat necesar. Nu trebuie să mai spunem că urmările sociale ale acestei scheme sunt imense. Este nostim că în final industria are grijă de securitate, dar este lamentabil că întreaga atenţie este concentrată asupra aplicării legii dreptului de autor şi nu asupra viruşilor, spărgătorilor, intruşilor şi altor probleme de securitate de care sunt interesate cele mai multe persoane.

Pe scurt, legiuitorii şi avocaţii vor fi ocupaţi în anii următori să pună echilibreze interesele economice ale proprietarilor de drepturi de autor şi interesul public. Cyber-spaţiul nu este diferit faţă de piaţa cărnii: în mod constant ea asmute un grup asupra celuilalt, de aici rezultând lupte pentru putere, litigii şi (sperăm) în final o urmă de reconciliere, cel puţin până va apare o nouă tehnologie diversionistă.

8.11REZUMAT

Criptografia este o unealtă ce poate fi folosită pentru menţinerea confidenţialităţii informaţiei şi pentru a asigura integritatea şi autenticitatea sa. Toate sistemele criptografice moderne sunt bazate pe principiul lui Kerckhoff de

avea un algoritm cunoscut public şi o cheie secretă. Mulţi algoritmi de criptografici folosesc transformări complexe, substituţii şi permutări pentru a transforma textul simplu în text cifrat. Totuşi, dacă criptografia cuantică poate fi pusă în practică, folosirea unor chei acoperitoare poate fi într-adevăr un sistem criptografic imposibil de spart.

Algoritmii criptografici pot fi divizaţi în algoritmi cu chei simetrice şi cu chei publice. Algoritmii cu chei simetrice amestecă biţii într-o serie de runde parametrizate de cheie pentru a schimba textul simplu în text cifrat. DES triplu şi Rijndael (AES) sunt cei mai cunoscuţi algoritmi cu cheie simetrică în momentul de faţă.

Algoritmii cu chei publice au proprietatea că sunt folosite diferite chei pentru criptare şi decriptare şi că cheia de decriptare nu poate fi derivată din cheia de criptare. Aceste proprietăţi fac posibilă publicarea cheii publice. Principalul algoritm cu cheie publică este RSA, care îşi trage puterea din faptul că este foarte dificil să se factorizeze numerele mari.

Documentele legale, comerciale şi altele au nevoie de semnături. Ca urmare, au fost propuse diverse scheme pentru semnăturile digitale, folosind algoritmi cu chei simetrice şi cu chei publice. Uzual, mesajele ce trebuie semnate sunt rezumate folosind algoritmi precum MD5 şi SHA-l şi apoi sunt semnate rezumatele şi nu mesajele originale.

Gestiunea cheilor publice poate fi făcută folosind certificate, care sunt documente ce leagă un principal de o cheie publică. Certificatele sunt semnate de către o autoritate de încredere sau de către cineva aprobat (recursiv) de o autoritate de încredere. Rădăcina lanţului trebuie obţinută înainte, dar programele de navigare au incluse multe certificate de rădăcini în ele.

Aceste instrumente criptografice pot fi folosite pentru a securiza traficul în reţea. IPsec operează la nivelul reţea, criptând fluxurile de pachete de la o gazdă la alta. Zidurile de protecţie pot verifica traficul ce iese sau intră într-o organizaţie, deseori pe baza protocolului şi a portului folosit. Reţelele private virtuale pot simula o veche reţea închiriată pentru a asigura proprietăţile necesare de securitate. În final, reţelele fără fir au nevoie de o bună securitate pe care WEP 802.11 nu o aduce, deşi 802.11i ar trebui să îmbunătăţească lucrurile destul de mult.

Când două părţi stabilesc o sesiune, trebuie să se autentifice una către alta şi dacă este nevoie să stabilească o cheie comună a sesiunii. Există diverse protocoale de autentificare, inclusiv câteva ce folosesc o terţă parte verificată, Diffie-Hellman, Kerberos şi criptografia cu cheie publică.

Securitatea mesajelor electronice poate fi obţinută printr-o combinaţie de tehnici pe care le-am studiat în cadrul acestui capitol. De exemplu, PGP comprimă mesajul, apoi îl criptează folosind IDEA. El trimite cheia IDEA criptată cu cheia publică a receptorului. În plus, rezumă mesajul şi trimite rezumatul semnat pentru verificarea integrităţii mesajului.

Securitatea pe Web este de asemenea un subiect important, începând cu securitatea numelor. DNSsec oferă o cale pentru prevenirea păcălirii DNS-ului, aşa cum fac şi numele cu auto-certificare. Majoritatea paginilor Web de comerţ

electronic folosesc SSL pentru a stabili sesiuni securizate, autentificate între client şi server. Diverse tehnici sunt folosite pentru a rezolva probleme legate de codul mobil, în special depunerea într-un mediu protejat şi semnarea codului.

Internet-ul ridică multe probleme în care tehnologia interacţionează puternic cu politica publică. Câteva dintre zone includ confidenţialitatea, libertatea de exprimare şi dreptul de autor.

8.12PROBLEME

1. Spargeţi următorul cifru monoalfabetic. Textul în clar, constând numai din litere, este un fragment dintr-o poezie a lui Lewis Carroll.

kfd ktbd fzm eubd kfd pzyiom mztx ku kzyg ur bzha kfthcm ur mfudm zhx mftnm zhx mdzythc pzq ur ezsszcdm zhx gthcm zhx pfa kfd mdz tm sutyhc fuk zhx pfdkfdi ntcm fzld pthcm sok pztk z stk kfd uamkdim eitdx sdruid pd fzld uoi efzk rui mubd ur om zid uok ur sidzkf zhx zyy ur om zid rzk hu foiia mztx kfd ezindhkdi kfda kfzhgdx ftb boef rui kfzk

2. Spargeţi următorul cifru de transpoziţie pe coloane. Textul este luat dintr-o carte despre calculatoare, astfel încât „computer” este un cuvânt probabil. Textul constă numai din litere (fără spaţii). Pentru claritate, textul cifrat este împărţit în blocuri de 5 caractere.

aauan cvlre rurnn dltme aeepb ytust iceat npmey iicgo gorch srsoc nntii imiha oofpa gsivt tpsit lborl otoex

3. **G**ăsiţi o cheie acoperitoare de 77 de biţi ce generează textul „Donald Duck” din textul cifrat din fig. 8-4.

4. Criptografia cuantică necesită un tun fotonic ce poate, la cerere, să tragă un singur foton ce conţine un singur bit. În această problemă, calculaţi câţi fotoni cară un bit pe o legătură de fibră de 100 Gbps. Presupuneţi că lungimea unui foton este egală cu lungimea lui de undă, care pentru scopul acestei probleme este de 1 micron. Viteza luminii în fibră este de 20 cm/ns.

5. Dacă Trudy capturează şi regenerează fotonii când se foloseşte criptografia cuantică, ea va amesteca o parte din ei şi va provoca erori în cheia acoperitoare a lui Bob. Ce fracţiune din biţii din cheia acoperitoare a lui Bob vor fi eronaţi, în medie?

6. **U**n principiu criptografic fundamental spune că toate mesajele trebuie să aibă redundanţă. Dar de asemenea ştim că redundanţa ajută un intrus să-şi verifice cheia ghicită. Consideraţi două forme de redundanţă. În primul caz, cei *n* biţi iniţiali ai textului simplu conţin un tipar cunoscut. În al doilea, n biţi de al sfârşitul mesajului conţin un rezumat a mesajului. Din punct de vedere al securităţii, sunt acestea două echivalente? Discutaţi răspunsul dumneavoastră.

7. În fig. 8-6, blocurile P şi S alternează. Deşi acest aranjament este agreabil, nu

este mai sigur să avem mai întâi toate blocurile P şi apoi toate blocurile S?

8. Proiectaţi un atac la DES bazat pe faptul că textul constă numai din litere mari ASCII, plus spaţiu, virgulă, punct şi virgulă, întoarcere la capăt (CR) şi linie nouă (LF). Nu se ştie nimic despre biţii de paritate ai textului.

9. În text am calculat că o maşină de spart cifruri cu un miliard de procesoare ce ar putea analiza o cheie într-o picosecundă ar lua numai 10^{10} ani pentru a sparge versiunea AES de 128 de biţi. Totuşi, maşinile curente ar putea avea 1024 de procesoare şi le-ar trebui 1 ms pentru a analiza o cheie, aşa că ne trebuie un factor de îmbunătăţire a performanţei de 10^{15} pentru a obţine maşina ce sparge AES. Dacă legea lui Moore (puterea de procesare se dublează la fiecare 18 luni) se menţine în vigoare, de câţi ani este nevoie pentru a putea construi maşina?

10. AES suportă chei de 256 de biţi. Câte chei are AES-256? Verificaţi dacă găsiţi numere în fizică, chimie, astronomie care au aceeaşi dimensiune. Folosiţi Internet-ul pentru a vă facilita căutarea de numere mari. Trageţi o concluzie din cercetarea dumneavoastră.

11. Să presupunem că un mesaj a fost criptat folosind DES cu înlănţuirea blocurilor cifrate. Un bit al textului cifrat în blocul *Q* este transformat accidental din 0 în 1 în timpul transmisiei. Cât de mult text va fi deformat ca urmare a acestui fapt?

12. Să considerăm din nou înlănţuirea blocurilor cifrate. În loc să transformăm un bit 0 în 1, un bit 0 este inserat în fluxul textului cifrat după blocul *C.* Cât de mult text va fi deformat?

13. Comparaţi înlănţuirea blocurilor cifrate cu modul cu reacţie cifrată în funcţie de numărul de operaţii de criptare folosite pentru transmiterea unui fişier mare. Care este mai eficient şi cu cât?

14. Folosind sistemul de criptare cu chei publice RSA cu a= 1, *b=2* etc.,

dacă*p=7* şi *q=11,* listaţi 5 valori permise pentru *d.*
Dacă *p*=13, *q*=31 şi *d*=7, cât este *e?*
Folosind *p*=5, *q*=11 şi *d*=27, găsiţi valoarea lui *e* şi criptaţi „abcdefghij”.

15. Să presupunem că un utilizator, Maria, descoperă că cheia sa privată RSA (*d*1, *n1)* este aceeaşi cu cheia publică RSA (*e*2, *n*2) a altui utilizator, Frances. Cu alte cuvinte, *d*1 = *e*2 şi *n1* = *n2.* Ar trebui să se gândească să-şi schimbe cheile sale publică şi privată? Explicaţi răspunsul dumneavoastră.

16. **Să considerăm** folosirea modului contor, cum este arătat în fig. 8-15, dar cu *IV* = 0. Folosirea lui 0 ameninţă securitatea cifrului în general?

17. Protocolul de semnătură din fig. 8-18 are următoarele puncte slabe. Dacă Bob se defectează, el poate pierde conţinutul RAM-ului propriu. Ce probleme pot apărea şi cum pot fi prevenite?

18. În fig. 8-20, putem vedea cum Alice poate trimite lui Bob un mesaj semnat. Dacă

Trudy înlocuieşte *P,* Bob poate să-l detecteze. Dar ce se întâmplă dacă Trudy înlocuieşte ambii *P* şi semnătura?

19. Semnăturile digitale au o slăbiciune potenţială datorită utilizatorilor leneşi. În tranzacţiile din comerţul electronic, un contract poate fi redactat şi utilizatorului i se cere să semneze dispersia sa SHA-1. Dacă utilizatorul nu verifică dacă contractul şi dispersia corespund, utilizatorul ar putea să semneze din neatenţie un alt contract. Să presupunem că Mafia încearcă să exploateze această slăbiciune pentru a câştiga ceva bani. Vor face o pagina Web cu plată (de exemplu, pornografie, jocuri de noroc, etc.) şi va cere noilor clienţi numărul cărţii de credit. Apoi, ei trimit un contract în care spun că clientul doreşte să folosească serviciile lor şi să plătească cu carte de credit şi roagă clientul să semneze, ştiind că cei mai mulţi vor semna fără să verifice dacă contractul şi rezumatul corespund. Arătaţi cum poate Mafia să cumpere diamante de la bijutier legitim pe Internet şi acestea să fie plătite de către clienţii nesuspicioşi.

20. O clasă de matematică are 20 de studenţi. Care este probabilitatea ca cel puţin doi stu- denţi să aibă aceeaşi zi de naştere? Presupuneţi că nimeni nu s-a născut pe 29 februarie, astfel că sunt 365 de zile de naştere posibile.

21. După ce Ellen s-a confesat lui Marilyn că a păcălit-o în ceea ce-l priveşte pe Torn, Marilyn a reuşit să evite această problemă dictând mesajele ulterioare unei maşini de dictat, şi punând noua secretară să le introducă. Marilyn a plănuit să examineze mesajele de pe terminalul ei după ce au fost introduse pentru a fi sigură că acestea conţin propriile cuvinte. Poate noua secretară să folosească atacul zilei de naştere, ca să falsifice un mesaj şi dacă da, cum? *Indicaţie:* Poate.

22. Consideraţi încercarea nereuşită a lui Alice de a obţine cheia publică a lui Bob în fig. 8-23. Presupuneţi că Bob şi Alice au în comun o cheie secretă, dar Alice tot vrea cheia publică a lui Bob. Există un mod prin care ea poate fi obţinută într-un mod securizat? Dacă da, cum?

23. Alice doreşte să comunice cu Bob, folosind criptografia cu chei publice. Ea stabileşte o conexiune cu cineva sperând că acesta este Bob. Ea îi cere cheia lui publică şi el i-o trimite în text clar împreună cu un certificat X.509 semnat de către rădăcina CA. Alice are deja cheia publică a rădăcinii CA. Ce paşi trebuie să îndeplinească Alice pentru a vedea că vorbeşte cu Bob? Presupuneţi că lui Bob nu-i pasă cu cine vorbeşte (de exemplu, Bob este un fel de serviciu public).

24. Presupuneţi că un sistem foloseşte PKI bazată pe o ierarhie cu structură de arbore de CA-uri. Alice doreşte să comunice cu Bob şi primeşte un certificat de la Bob semnat de CA *X* după ce s-a stabilit un canal de comunicaţie cu Bob. Să considerăm că Alice nu a auzit niciodată de *X*. Ce paşi trebuie urmaţi de Alice pentru a verifica dacă vorbeşte cu Bob ?

25. Se poate ca IPsec cu AH să fie folosit în modul transport de una dintre maşinile ce se află în spatele unei cutii NAT ? Explicaţi răspunsul.

26. Precizaţi un avantaj al folosirii HMAC faţă de folosirea RSA pentru semnarea

rezumatelor SHA-1.'

27. Daţi un motiv pentru care un zid de protecţie ar putea fi configurat să cerceteze traficul dinspre exterior. Daţi un motiv pentru care ar putea fi configurat să inspecteze traficul spre exterior. Credeţi că aceste inspecţii au şanse de succes ?

28. Formatul pachetului WEP este arătat în fig. 8-31. Să presupunem că suma de control este de 32 de biţi, calculată prin SAU ECLUSIV asupra tuturor cuvintelor de 32 de biţi din încărcătura utilă luate împreună. De asemenea să presupunem că problemele cu RC4 sunt corectate prin înlocuirea cu un cifru-şir ce nu are slăbiciuni şi că toate elementele IV sunt extinse la 128 de biţi. Există vreo cale pentru ca un intrus să spioneze sau să interfereze cu traficul fără a fi detectat?

29. Să presupunem că o organizaţie ce foloseşte VPN pentru a conecta securizat siturile sale din Internet. Este nevoie ca un utilizator din această organizaţie să folosească criptarea sau alt mecanism de securitate pentru a comunica cu alt utilizator din cadrul organizaţiei?

30. Modificaţi un mesaj în protocolul din fig. 8-34 pentru a-l face rezistent la atacurile prin reluare. Explicaţi de ce funcţionează modificarea.

31. Schimbul de chei Diffie-Hellman este folosită pentru stabilirea unei chei secrete între Alice şi Bob. Alice trimite lui Bob (719, 3, 191). Bob răspunde cu (543). Numărul secret al lui Alice este x = 16. Care este cheia secretă?

32. Dacă Alice şi Bob nu s-au întâlnit niciodată, nu au secrete comune, şi nu au certificate, ei totuşi pot să stabilească o cheie secretă comună folosind algoritmul Diffie-Hellman. Explicaţi de ce le este foarte greu să se apere împotriva unui atac omul-din-mijloc.

33. În protocolul din fig. 8-39, de ce este A trimis în clar împreună cu cheia de sesiune criptată?

34. În protocolul din fig. 8-39, am arătat că a începe fiecare mesaj transmis în clar cu 32 de biţi zero este riscant. Să presupunem că fiecare mesaj începe cu un număr aleatoriu al utilizatorului, care este practic o a doua cheie secretă cunoscută doar de utilizatorul ei şi de KDC. Se elimină în acest mod atacul textului clar cunoscut? De ce?

35. În protocolul Needham-Schroeder, Alice generează 2 provocări, R_A şi R_{A2}. Aceasta seamănă cu o exagerare. Nu ar fi fost suficientă una singură?

36. Să presupunem că o organizaţie foloseşte Kerberos pentru autentificare. Ce se întâmplă, din punctul de vedere al securităţii şi al disponibilităţii serviciului, dacă AS sau TGS cad?

37. În protocolul de autentificare cu chei publice din fig. 8-43, în mesajul 7, R_B este criptat cu K_S. Este această criptare necesară, sau ar fi fost potrivit să fie trimisă ca text clar? Explicaţi răspunsul.

38. Terminalele din punctele de vânzare care folosesc cartele cu bandă magnetică şi coduri PIN au un defect fatal: un negustor răuvoitor poate modifica cititorul de coduri propriu pentru a captura şi a memora toată informaţia de pe cartelă precum şi codul PIN pentru a transmite ulterior tranzacţii suplimentare (falsificate). Următoarea generaţie de terminale din punctele de vânzare va folosi cartele cu unităţi centrale complete, tastatură şi monitor pe cartelă. Implementaţi un protocol pentru acest sistem, pe care negustorii răuvoitori să nu-l poată sparge.

39. Enumeraţi două motive pentru care PGP comprimă mesajele.

40. Presupunând că toată lumea de pe Internet foloseşte PGP, ar putea un mesaj PGP să fie trimis la o adresă Internet arbitrară şi să fie decodificat corect de toţi cei implicaţi? Discutaţi răspunsul.

41. Atacului prezentat în fig. 8-47 îi lipseşte un pas. Acest pas nu este necesar pentru ca atacul să funcţioneze, dar includerea lui ar putea reduce eventualele suspiciuni ulterioare. Care este pasul lipsă?

42. S-a propus împiedicarea păcălirii DNS-ului folosind predicţia identificatorilor, metodă în care serverul pune un identificator aleatoriu în loc să utilizeze un contor. Discutaţi aspectele legate de securitate ale acestei abordări.

43. Protocolul de transport de date al SSL implică două numere ad-hoc şi o cheie primară. Ce rol are folosirea numerelor ad-hoc (dacă are vreunul)?

44. Imaginea fin fig. 8-55(b) conţine textul ASCII a cinci piese de Shakespeare. Ar fi posibil să fie ascunsă muzică printre zebre în loc de text? Dacă da, cum ar funcţiona şi cât de mult aţi putea ascunde în acea imagine? Dacă nu, de ce nu?

45. Alice era o utilizatoare fidelă a unui retransmiţător anonim de tip 1. Ea trimitea multe mesaje grupului ei de ştiri preferat, *alt.fanclub.alice* şi toată lumea ştia că acestea vin de la Alice pentru că toate purtau acelaşi pseudonim. Presupunând că retransmiţătorul funcţiona corect, Trudy nu putea să pretindă că e Alice. După ce toate retransmiţătoarele de tip 1 au fost desfiinţate, Alice s-a mutat la unul care utilizează criptografia şi a început o nouă serie de mesaje în grupul ei. Imagi- naţi-vă un mod de a o opri pe Trudy să trimită mesaje noi grupului, pretinzând că este Alice.

46. Căutaţi pe Internet nu caz interesant implicând confidenţialitatea şi scrieţi o prezentare de o pagină despre acesta.

47. Căutaţi pe Internet un caz ajuns la tribunal, implicând dreptul de autor şi utilizarea corectă şi scrieţi o prezentare de o pagină, rezumând ceea ce aţi găsit.

48. Scrieţi un program care criptează datele de intrare, aplicându-le operaţia XOR cu un şir-cheie. Găsiţi sau scrieţi şi un generator de numere aleatorii pentru a putea genera şirul-cheie. Programul ar trebui să se comporte ca un filtru, preluând text clar de la intrarea standard şi producând text cifrat la ieşirea

standard (şi invers). Programul trebuie să aibă un singur parametru, cheia de la care porneşte generatorul de numere aleatoare.

49. Scrieţi o procedură care calculează rezumatul unui bloc de date folosind SHA-1. Procedura trebuie să aibă doi parametri: o referinţă la zona tampon de intrare şi o referinţă la o zonă tampon, de 20 de octeţi, de ieşire. Pentru a vedea specificaţiile exacte ale SHA-l, căutaţi pe Internet FIPS 180-l, care este specificaţia completă.

RECOMANDĂRI DE LECTURĂ ŞI BIBLIOGRAFIE

Aici se încheie studiul nostru despre reţelele de calculatoare, dar această lucrare este doar un început. Multe subiecte interesante nu au fost tratate la nivelul de detaliu pe care l-ar fi meritat, iar altele au fost omise în totalitate, din lipsă de spaţiu. Pentru cititorii care doresc să continue studiul reţelelor de calculatoare furnizăm în acest capitol câteva sugestii de lecturi posibile, precum şi o listă bibliografică.

9.1 SUGESTII PENTRU LECTURI VIITOARE

Există o literatură vastă despre toate aspectele reţelelor de calculatoare. Trei reviste, care publică frecvent articole în acest domeniu sunt *IEEE Transactions on Communications, IEEE Journal on Selected Areas în Communications* şi *Computer Communication Review* . De asemenea, multe alte reviste publică ocazional articole cu acest subiect.

IEEE mai publică şi trei reviste ilustrate – *IEEE Internet Computing, IEEE Network Magazine* şi *IEEE Communications Magazine* – care conţin studii, îndrumări practice, studii de caz despre reţele. Primele două reviste se referă cu

precădere la arhitectură, standarde şi software, iar ultima se orientează către tehnologia comunicaţiilor (fibră optică, sateliţi şi aşa mai departe).

În plus, există un număr de conferinţe anuale sau bianuale care atrag multe lucrări despre reţele şi sisteme distribuite, în particular *SIGCOMM Annual Conference, The International Conference on Distributed Computer Systems* şi *The Symposium on Operating Systems Principles*

Enumerăm în continuare, în ordinea capitolelor cărţii, câteva sugestii de lecturi suplimentare. Cele mai multe dintre acestea sunt îndrumări practice sau studii asupra subiectelor la care se referă. Câteva dintre ele sunt capitole din diverse manuale.

9.1.1 Lucrări introductive şi generale

Bi et. al, „Wireless Mobile Communications at the Start of the 21st Century”

Un nou secol, o nouă tehnologie – un subiect incitant. După un istoric al comunicaţiilor fără fir, sunt tratate aspectele de bază, inclusiv standarde, aplicaţii, Internet şi tehnologie.

Comer, *The Internet Book*

Cine caută o introducere simplă în Internet ar trebui să citească această lucrare. Comer descrie istoria, dezvoltarea, tehnologia, protocoalele şi serviciile Internet-ului în termeni pe care novicii îi pot înţelege, dar datorită cantităţii de material acoperite, cartea sa prezintă interes şi pentru cititorii mai avansaţi.

Garber, „Will 3G Really Be the Next Big Wireless Technology?”

De la telefoanele mobile din a treia generatie se aşteaptă să fie capabile de transmisii de voce şi de date la rate de transfer de până la 2 Mbps, însă startul acestei tehnologii a fost unul lent. Promisiunile, obstacolele, tehnologia, considerentele politice şi economice ale utilizării comunicaţiei wireless în bandă largă sunt tratate în acest articol uşor de citit.

IEEE Internet Computing, Ianuarie – Februarie 2000

Primul număr al revistei *IEEE Internet Computing* din noul mileniu a apărut exact în formatul pe care l-am fi aşteptat: personalităţi care în mileniul precedent au contribuit la crearea Internetului au fost invitate să-şi expună viziunile asupra evoluţiei Internetului în mileniul următor. Printre experţi se numără Paul Baran, Lawrence Roberts, Leonard Kleinrock, Stephen Crocker, Danny Cohen, Bob Metcalfe, Bill Gates, Bill Joy. Pentru a valorifica la maxim acest material, este recomandat să aşteptaţi 500 de ani şi *apoi* să citiţi profeţiile lor.

Kipnis, „Beating the System: Abuses of the Standards Adoption Process”

Comisiile care se ocupă cu adoptarea standardelor încearcă să menţină o poziţie neutră în demersurile lor, însă din nefericire există companii care încearcă să abuzeze de sistem. De exemplu, s-au înregistrat în repetate rânduri cazuri în care după adoptarea unui standard, o anumită companie care a participat la dezvoltarea acestuia să anunţe ca deţine un brevet ce stă la baza standardului şi să acorde licenţe doar anumitor companii agreate şi la preţuri stabilite doar după criterii proprii. Articolul este un început excelent în studiul „părţii întunecate” a

procedurilor de adoptare a standardelor.

Kyas şi Crawford, *ATM Networks*

ATM, cotat la un moment dat ca protocolul de reţea al viitorului, joacă încă un rol important în cadrul sistemului telefonic. Această lucrare se constituie într-un ghid la zi, conţinând informaţii detaliate despre protocoalele ATM şi despre modul de integrare al acestora cu reţele bazate pe protocolul IP.

Kwok, „A Vision for Residential Broadband Service”

Dacă doriţi să aflaţi viziunea Microsoft asupra serviciului de video la cerere în anul 1995, citiţi acest articol. Cinci ani mai târziu această viziune era complet depăşită. Valoarea acestui articol constă în a demonstra că până şi persoane puternic motivate şi cu un nivel înalt de cunoştinţe în domeniu se găsesc în imposibilitatea de a estima cu precizie evoluţia chiar şi pe un termen de numai 5 ani. Ar trebui să fie o lecţie pentru noi toţi.

Naughton, *A Brief History of the Future*

De fapt, cine este inventatorul Internetului? Multe persoane îşi asumă acest merit, şi pe bună dreptate, din moment ce mulţi au contribuit în diferite moduri la crearea sa. Acest scurt istoric al Internetului arată cum s-au întâmplat lucrurile într-o manieră spirituală şi plăcută, presărată cu anecdote, cum ar fi convingerea AT&T, exprimată în mod repetat, că nu există nici un viitor în domeniul comunicaţiilor digitale.

Perkins, „Mobile Networking în the Internet”

Pentru a vă forma o imagine de ansamblu asupra protocoalelor din domeniul reţelelor mobile, nivel cu nivel, aceasta este lucrarea pe care trebuie să o consultaţi. Sunt examinate rând pe rând nivelurile de la fizic până la transport, precum şi middleware-ul, considerente de securitate şi reţelele ad-hoc.

Teger şi Waks, „End-User Perspectives on Home Networking”

Reţelele pentru utilizatori individuali nu se aseamănă cu cele pentru corporaţii. Aplicaţiile sunt diferite (preponderent multimedia), echipamentele provin de la o gamă largă de producători iar utilizatorii au un nivel scăzut de pregătire tehnică şi o toleranţă mică la defecţiuni de funcţionare. Citiţi această carte pentru a afla mai multe.

Varshney şi Vetter, „Emerging Mobile şi Wireless Networks”

Acesta este o altă introducere în domeniul comunicaţiilor fără fir. Sunt discutate LAN-urile fără fir, buclele locale fără fir, sateliţii precum şi unele aspecte legate de software-ul şi aplicaţiile aferente.

Wetteroth, *OSI Reference Model for Telecommunications*

Deşi protocoalele stivei OSI nu mai sunt utilizate ca atare, modelul cu şapte niveluri a căpătat o mare popularitate. Lucrarea oferă mai multe explicaţii referitoare la modelul OSI şi îşi propune să aplice acest model reţelelor de telecomunicaţii (în contrast cu reţelele de calculatoare), evidenţiind unde se încadrează în stiva OSI diverse protocoale de telefonie şi transmisie de voce.

9.1.2 Nivelul fizic

Abramson, „Internet Access Using VSATs”

Staţiile terestre de dimensiune redusă devin din ce în ce mai utilizate atât pentru sistemele de telefonie din mediul rural cât şi pentru accesul Internet al corporaţiilor din ţările dezvoltate. Totuşi, natura traficului diferă radical în cele două situaţii, aşa că sunt necesare protocoale diferite. În acest articol, inventatorul protocolului ALOHA discută despre numeroase metode de alocare a benzii de transmisie utilizate în sisteme VSAT.

Alkhatib et. al, „Wireless Data Networks: Reaching the Extra Mile”

Această lucrare, concepută ca un îndrumar practic, este un bun punct de pornire pentru o introducere rapidă în terminologia şi tehnologia utilizată de reţelele fără fir.

Azzam şi Ransom, *Broadband Access Technologies*

Sistemul de telefonie, fibrele optice, ADSL, reţelele de televiziune prin cablu, sateliţi, chiar şi liniile de înaltă tensiune ca tehnologii de acces în reţea sunt discutate în această lucrare. Alte subiecte includ reţele pentru utilizatori individuali, servicii, performanţele reţelelor şi standarde. Lucrarea se încheie cu biografiile marilor companii din industria telecomunicaţiilor şi reţelelor, dar luând în considerare evoluţia rapidă a acestui domeniu, acest capitol va avea probabil o existenţă mai scurtă decât capitolele referitoare la tehnologii.

Bellamy, *Digital Telephony*

Această lucrare demnă de încredere conţine tot ce aţi dorit vreodată să ştiţi despre sistemul telefonic şi chiar mai mult. În particular, sunt interesante capitolele despre transmisie şi multiplexare, comutare digitală, fibră optică, telefonie mobilă şi DSL .

Berezdivin et. al, „Next-Generation Wireless Communications Concepts and Technologies”

Autorii sunt cu un pas înaintea tuturor celorlalţi. „Următoarea generaţie” din titlu se referă la a patra generaţie de comunicaţii fără fir. De la aceste reţele se aşteaptă să furnizeze servicii IP în orice loc, o legătură ireproşabilă la Internet cu lăţime mare de bandă şi o calitate excelentă a serviciilor. Aceste scopuri vor fi atinse prin alocarea inteligentă a spectrului, alocarea dinamică a resurselor şi servicii adaptabile. Este o viziune relativ îndepărtată, dar acelaşi lucru se putea spune în 1995 şi despre telefonia mobilă .

Dutta-Roy, „An Overview of Cable Modem Technology and Market Perspectives”

Televiziunea prin cablu a evoluat de la simplul sistem CATV la sisteme complexe de distribuţie pentru televiziune, Internet şi telefonie. Aceste schimbări au afectat în mare măsură şi infrastructura de cablu. Acest articol merită citit pentru o dezbatere asupra fabricilor de cablu, a standardelor, a marketing-ului cu accentul căzând pe DOCSIS .

Farserotu şi Prasad, „A Survey of Future Broadband Multimedia Satellite

Systems, Issues, and Trends”

O gamă largă de sateliţi de comunicaţii de date se află pe orbită sau pe planşele de proiectare, incluzând Astrolink, Cyberstar, Spaceway, Skybridge, Teledesic şi iSk. Aceştia utilizează diverse tehnici printre care conductă curbă şi comutare prin sateliţi. Pentru o imagine de ansamblu a diferitelor tehnici şi sisteme de sateliţi această lucrare este un bun punct de pornire.

Hu şi Li, „Satellite-Based Internet: A Tutorial”

Accesul la Internet prin satelit diferă de accesul prin liniile terestre. Nu numai că apare o problemă legată de întârziere, dar şi rutarea şi comutarea sunt diferite. În această lucrare, autorii dezbat câteva dintre problemele legate de accesul Internet prin intermediul sateliţilor.

Joel, „Telecommunications and the IEEE Communications Society”

Acest articol oferă un istoric compact dar cuprinzător al telecomunicaţiilor, începând cu telegraful şi încheind cu standardul 802.11. Totodată, sunt tratate aspecte legate de radio, telefonie, comutare analogică şi digitală, cabluri submarine, transmisii digitale, ATM, staţii de televiziune, televiziune prin cablu, comunicaţii prin fibră optică, telefonie mobilă, comutare de pachete, ARPANET şi Internet.

Metcalfe, „Computer/Network Interface Design Lessons from Arpanet & Ethernet”

Deşi inginerii au construit interfeţe de reţea începând în urmă cu zeci de ani, unii se întreabă încă dacă au învăţat ceva din această experienţă. În acest articol, proiectantul Ethernet-ului spune cum se

construieşte o interfaţă de reţea şi ce se face cu ea odată ce a fost construită. El recunoaşte sincer ce a făcut greşit şi ce a făcut corect.

Palais, Fiber Optic Communication, ediţia a 3-a

Deşi cărţile despre tehnologia fibrei optice devin din ce în ce mai specializate, această lucrare este mai accesibilă decât multe altele. Ea acoperă ghidurile de undă, sursele de lumină, detectoarele de lumină, cuploarele, modularea, zgomotul şi multe alte subiecte.

Pandya, „Emerging Mobile and Personal Communications Systems”

Acest articol este foarte potrivit ca o introducere scurtă şi plăcută în sistemele de comunicaţie portabile personale. Una din cele nouă pagini conţine o listă de 70 de acronime folosite în celelalte opt pagini.

Sarikaya, „Packet Mode în Wireless Networks: Overview of Transition of Third Generation”

Conceptul de bază a reţelelor celulare din generaţia a treia constă în transmisii de date fără fir. Acest articol este foarte potrivit pentru a obţine o imagine de ansamblu asupra transmisiilor de date în reţelele din generaţia a doua şi cum vor evolua acestea în reţelele din generaţia a treia. Sunt tratate subiecte ca GPRS, IS-95B, WCDMA şi CDMA 2000.

9.1.3 Nivelul legătură de date

Carlson, PPP Design, Implementation and Debugging, editia a 2-a

Dacă vă interesează informaţii detaliate despre toate protocoalele ce formează suita PPP, inclusiv CCP (compresie) şi ECP (criptare), acestă lucrare este de referinţă. Se pune un accent deosebit pe ANU PPP-2.3, o implementare larg răspândită a PPAG.

Gravano, *Introduction to Error Control Codes*

Erori se strecoară în aproape toate formele de comunicaţie digitală; din acest motiv au fost dezvoltate multe tipuri de coduri detectoare şi corectoare de erori. Această lucrare descrie unele dintre cele mai importante coduri, de la de la Codul Hammimg până la codul Galois şi codurile convoluţionale, necesitând un volum minim de cunoştinţe de algebră din partea cititorului.

Holzmann, Design and Validation of Computer Protocols

Cititorii interesaţi în aspectele mai formale ale protocoalelor legăturii de date (şi similare) ar trebui să vadă această carte. Sunt prezentate aici specificarea, modelarea, corectitudinea şi testarea acestor protocoale.

Peterson şi Davie, *Computer Networks: A System Approach*

Capitolul 2 tratează multe aspecte legate de nivelul legatură de date, inclusiv încadrarea (framing), protocoalele start-stop, protocoale cu fereastră glisantă şi LAN-urile IEEE 802.

Stallings, *Data and Computer Communications*

Capitolul 7 tratează aspecte legate de nivelul legatură de date cum ar fi controlul fluxului, detectarea erorilor şi protocoale simple de nivel legatură de date, inclusiv start-stop şi protocolul cu revenire cu n paşi. Sunt discutate şi protocoalele de tip HDLC.

9.1.4 Subnivelul de control al accesului la mediu

Bhagwat, „Bluetooth: Technology for Short-Range Wireless Apps”

Lucrarea reprezintă un bun punct de pornire pentru o introducere directă a sistemului Bluetooth. Sunt discutate protocoalele şi profilurile de bază, radio, picoreţele şi conexiuni, urmate de o introducere a diverselor protocoale.

Bisdikian, „An Overview of the Bluetooth Wireless Technology”

La fel ca lucrarea precedentă a lui Bhagwat, şi aceasta este un bun punct de pornire pentru a afla totul despre sistemul Bluetooth. Sunt prezentate printre altele picoreţelele, stiva de protocoale şi profilurile.

Crow et. al, „IEEE 802.11 Wireless Local Area Networks”

Această lucrare este un bun punct de pornire pentru o introducere simplă în tehnologia şi protocoalele 802.11. Accentul cade pe subnivelul MAC. Sunt discutate atât controlul distribuit cât şi controlul centralizat. În încheiere sunt prezentate câteva evaluări ale performanţelor 802.11 în urma simulărilor realizate în condiţii

diverse.

Eklund et. al, „IEEE Standard 802.16: A Technical Overview of the Wireless MAN Air Interface for Broadband Wireless Access”

Standardul 802.16 adoptat de IEEE în 2002 presupune utilizarea comunicaţiei fără fir pentru buclele locale, ceea ce ar revoluţiona serviciile telefonice, aducând transmisia fără fir în bandă largă către locuinţe. În această privire de ansamblu, autorii expun principalele aspecte tehnologice ce ţin de acest standard.

Kapp, „802.11: Leaving the Wire Behind”

Această scurtă introducere a standardului 802.11 acoperă chestiunile de bază, protocoalele şi standardele relevante.

Kleinrock, „On Some Principles of Nomadic Computing and Multi-Access Communications”

Partajarea canalului de transmisie în medii neghidate este o problemă mai complexă decat partajarea canalului de transmisie în medii ghidate. În acest articol sunt expuse probleme ca topologii dinamice, rutare, consumul de energie precum şi alte aspecte legate de accesul la canalul de transmisie al echipamentelor mobile fără fir.

Miller şi Cummins, *LAN Technologies Explained*

Aveţi nevoie să cunoaşteţi cât mai multe despre tehnologiile care pot fi utilizate în LAN-uri? Această lucrare acoperă cele mai multe dintre aceste tehnologii, inclusiv FDDI şi token ring, precum şi larg răspânditul Ethernet. În timp ce primele două sunt rar utilizate în instalările noi de reţele, multe reţele deja existente le mai utilizează încă, iar reţelele cu topologie de inel sunt încă des întâlnite (de exemplu SONET).

Perlman, *Interconnections*, ediţia a 2-a

Lucrarea reprezintă o expunere demnă de încredere, dar totuşi distractivă, despre punţi, rutere şi rutare în general. Autorul este cel care a proiectat algoritmul “spanning tree” utilizat de punţile IEEE 802 şi este una dintre autorităţile de nivel mondial în ceea ce priveşte diverse aspecte ale reţelelor.

Webb, „Broadband Fixed Wireless Access”

Această lucrare răspunde la întrebările „de ce” şi „cum” în ceea ce priveşte transmisiile fără fir în bandă largă fixă. Secţiunea „de ce” argumentează că utilizatorii nu doresc adrese de e-mail separate pentru acasă şi pentru serviciu, numere diferite de telefon pentru acasă, la serviciu sau pentru telefonul mobil, un cont pentru discuţii (chat) şi un număr sau două de fax. În schimb, ei preferă un sistem unic, integrat, care să funcţioneze pretutindeni. Accentul în secţiunea referitoare la tehnologie cade asupra nivelului fizic şi sunt discutate subiecte ca număr de purtătoare, comparaţii între TDD şi FDD, între modulare fixă şi modulare adaptivă.

9.1.5 Nivelul reţea

Bhatti şi Crowcroft, „QoS Sensitive Flows: Issues în IP Packet Handling”

Una din modalităţile de a obţine o mai bună calitate a serviciilor într-o reţea este planificarea riguroasă a plecării pachetelor din fiecare ruter. În această lucrare este prezentată o serie de algoritmi de planificare a pachetelor precum şi alte detalii legate de acest subiect.

Chakrabarti, „QoS Issues în Ad Hoc Wireless Networks”

Rutarea în reţelele ad-hoc de calculatoare portabile care se află la distanţă mică unele de altele este destul de dificilă şi fără a ţine cont de calitatea serviciilor. Totuşi, utilizatorii sunt interesaţi de calitatea serviciilor aşa că trebuie acordată atenţie acestui aspect. Particularitatea reţelelor ad-hoc şi alte aspecte legate de rutare şi calitatea serviciilor sunt discutate în acest articol.

Comer, *Internetworking with TCP/IP,* Vol.1, ediţia a 4-a

Comer a scris cartea decisivă despre suita de protocoale TCP/IP. Capitolele de la 4 la 11 tratează IP şi protocoalele legate de el din nivelul reţea. Celelalte capitole se referă în primul rând la nivelele superioare şi merită, de asemenea, să fie citite.

Huitema, Routing în the Internet

Dacă vreţi să ştiţi tot ce este de ştiut despre dirijarea în Internet, aceasta este cartea care vă trebuie. Sunt trataţi în mare detaliu atât algoritmi care pot fi pronunţaţi (ca RIP, CIDR şi MBONE) cât şi algoritmi care nu pot fi pronunţaţi (ca OSPF, IGRP, EGP şi BGP). Se găsesc aici caracteristici noi, cum ar fi trimiterea cu destinaţie multiplă (multicast), mobil, IP şi rezervarea resurselor

Malhotra, *IP Routing*

Lucrarea conţine foarte multă informaţie, constituindu-se într-un îndrumar detaliat asupra rutării IP. Sunt discutate protocoale ca RIP, RIP-2, IGRP, EIGRP, OSPF şi BGP-4 .

Metz, „Differentiated Services”

Garanţiile oferite de conceptul de calitate a serviciilor sunt importante pentru multe aplicaţii multimedia. Serviciile integrate şi serviciile diferenţiate sunt două abordări posibile pentru obţinerea acestor garanţii. Ambele sunt discutate aici, cu accent pe serviciile diferenţiate.

Metz, „IP Routers: New Tool for Gigabit Networking”

Majoritatea referinţelor bibliografice pentru capitolul 5 se referă la algoritmi de rutare. Nu şi aceasta, care explică modul de funcţionare al ruterelor. Ruterele au evoluat de la staţii de lucru pentru uz general până la maşini dedicate pentru rutare. Dacă doriţi să aflaţi mai multe despre rutere, acest articol este un bun punct de pornire.

Nemeth s.a, *UNIX System Administration Handbook*

Pentru o schimbare de ritm, capitolul 13 din această lucrare abordează o viziune mai practică asupra reteţelor decât cele mai multe dintre celelalte referinţe bibliografice. În loc să discute concepte abstracte, ea oferă multe sfaturi despre

cum se administrează o reţea reală.

Perkins, „Mobile Networking through Mobile IP”

Pe măsură ce dispozitivele mobile de calcul devin din ce în ce mai populare, noţiunea de IP mobil devine din ce în ce mai importantă. Acest îndrumar practic oferă o introducere în acest domeniu discutând şi alte subiecte adiacente.

Perlman, *Interconnections: Bridges and Routers,* ediţia a 2-a

În capitolele de la 12 pana la 25, Perlman expune multe probleme ce apar în proiectarea algoritmilor de rutare unicast şi multicast, atât pentru WAN-uri cât şi pentru LAN-uri interconectate, precum şi implementările acestor algoritmi în diverse protocoale. Cea mai interesantă parte este însă capitolul 18, în care autoarea analizează, într-o manieră instructivă şi amuzantă, anii de experienţă în protocoale de reţea.

Puzmanova, *Routing and Switching: Time of Convergence?*

La sfârşitul anilor 1990, unii producători de echipamente de reţea au început să denumească orice drept comutator, în timp ce administratorii de reţele de mari dimensiuni afirmau că au început procesul de trecere de la rutere la comutatoare. Aşa cum sugerează şi titlul, lucrarea discută despre viitorul ruterelor şi comutatoarelor, întrebându-se dacă aceste două echipamente vor converge către un punct comun.

Royer şi Toth, „A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks”

Algoritmul de rutare ad-hoc AODV, prezentat în capitolul 5 nu este singurul cunoscut. O diversitate de alţi algoritmi, printre care DSDV, CGSR, WRP, DSR, TORA, ABR, DRP şi SRP sunt prezentaţi şi comparaţi în această lucrare. Evident, dacă vă propuneţi să inventaţi un nou algoritm de rutare ad-hoc, primul pas este alegerea unui acronim de trei litere pentru denumire.

Stevens, *TCP/IP Illustrated*, Vol. 1

Capitolele 3-10 furnizează o tratare cuprinzătoare, ilustrată prin exemple, a IP-ului şi a protocoalelor legate de el (ARP, RARP şi ICMP).

Striegel şi Manimaran, „A Survey of QoS Multicasting Issues”

Noţiunile de multicasting şi calitate a serviciilor capătă o importanţă din ce în ce mai mare pe măsură ce iau amploare servicii ca radioul şi televiziunea prin internet. În acest studiu, autorii discută modul în care algoritmii de rutare pot ţine cont de aceşti doi factori.

Yang şi Reddy, „A Taxonomy for Congestion Control Algorithms în Packet Switching Networks”

Autorii au proiectat o taxonomie pentru algoritmii de control al congestiei. Principalele categorii sunt buclă deschisă cu controlul sursei, buclă deschisă cu controlul destinaţiei, buclă închisă cu reacţie explicită şi buclă închisă cu reacţie implicită. Ei folosesc această taxonomie pentru a descrie şi clasifica 23 de algoritmi existenţi.

9.1.6 Nivelul transport

Comer, *Internetworking with TCP/IP,* Vol. 1, ediţia a 4-a

Aşa cum s-a mai menţionat, Comer a scris lucrarea decisivă despre suita de protocoale TCP/IP. Capitolul 12 se referă la UDP, iar capitolul 13 la TCP.

Hall şi Cerf, *Internet Core Protocols: The Definitive Guide*

Daca vă place să obţineţi informaţia direct de la sursă, atunci acesta este locul potrivit pentru a afla mai multe despre TCP. În fond, Cerf este unul dintre inventatorii TCP-ului. Capitolul 7 este o prezentare exactă a TCP, în care se arată cum trebuie interpretată informaţia furnizată de uneltele de analiză a protocoalelor şi management al reţelelor. Alte capitole expun aspecte legate de UDP, IGMP, ICMP şi ARP.

Kurose şi Ross, *Computer Networking: A Top-Down Approach Featuuring the Internet*

Capitolul 3 se referă la nivelul transport şi conţine o cantitate importantă de informaţie despre TCP şi UDP. Sunt discutate de asemenea protocoalele start-stop şi cu revenire cu n paşi examinate în capitolul 3.

Mogul, „IP Network Performance”

În ciuda titlului acestui articol, el tratează cel puţin, dacă nu ceva mai mult, performanţele TCP- ului şi ale reţelelor în general, apoi performanţele IP-ului în particular. Este plin de îndrumări utile şi reguli de aur.

Peterson şi Davie, *Computer Networks: A Systems Approach*

Capitolul 5 se referă la UDP, TCP şi alte câteva protocoale înrudite. Sunt tratate pe scurt şi aspecte legate de performanţele reţelelor.

Stevens, TCP/IP Illustrated, Vol. 1

Capitolele 17-24 furnizează o tratare cuprinzătoare a TCP-ului, ilustrată prin exemple.

9.1.7 Nivelul aplicaţie

Bergholz, „Extending Your Markup: An XML Tutorial”

O introducere scurtă şi directă în XML pentru începători.

Cardellini et. al, *The State-of-the-Art în Locally Distributed Web-Server Systems*

Pe măsură ce serviciul Web creşte în popularitate, unele situri Web sunt nevoite să utilizeze un număr mare de servere pentru a putea face faţă traficului. Partea dificilă în construirea unei ferme de servere o constituie distribuirea încărcării între maşini. Acest îndrumar practic acoperă acest aspect în detaliu.

Berners-Lee et. al, „The World Wide Web”

O perspectivă asupra Web-ului şi a direcţiei sale de evoluţie prezentată de persoana care l-a inventat şi de câţiva colegi de la CERN. Articolul se concentrează asupra arhitecturii Web, URL-uri, HTTP şi HTML, cât şi asupra direcţiilor de

dezvoltare viitoare.

Choudbury et. al, „Copyright Protection for Electronic Publishing on Computer Networks”

Deşi numeroase cărţi şi articole descriu algoritmi criptografici, puţine descriu modul în care aceştia pot fi folosiţi pentru a împiedica utilizatorii să redistribuie documentele pe care au dreptul să le decripteze. Acest articol descrie o varietate de mecanisme care poate ajuta protecţia drepturilor de autor în era elecrtronică.

Collins, „Carrier Grade Voice over IP”

Dacă aţi citit lucrarea scrisă de Varshney et. al şi doriti să aflaţi toate detaliile despre transmisia de voce peste IP utilizând protocolul H.323, acesta este locul potrivit. Deşi cartea este lungă şi cuprinde multe detalii, este concepută ca un îndrumar practic şi nu necesită cunoştinţe anterioare de ingineria sistemelor telefonice.

Davison, „A Web Caching Primer”

Pe măsură ce Web-ul creşte în dimensiuni, tehnica de ascundere (caching) devine esenţială pentru a obţine performanţe bune. Lucrarea se constituie într-o scurtă introducere în “Web caching”.

Krishnamurthy şi Rexford, *Web Protocols and Practice*

Este dificil să găseşti o lucrare mai cuprinzătoare despre toate aspectele Web-ului decât aceasta. Sunt tratate aspecte legate de clienţi, servere, proxy-uri şi ascundere. Există de asemenea capitole despre măsurători de trafic Web cât şi despre cercetările prezente în acest domeniu.

Rabinovich şi Spatscheck, *Web Caching and Replication*

Lucrarea este o investiţie bună pentru o tratare cuprinzătoare a conceptelor de “Web caching” şi replicare. Proxy-uri, cache-uri, încărcare, reţele cu livrare după conţinut, selecţia serverelor şi multe altele sunt prezentate în detaliu.

Shahabi et. al, „Yima: A Second-Generation Continuous Media Server”

Serverele multimedia reprezintă sisteme complexe ce trebuie să rezolve probleme ca planificarea procesoarelor, localizarea fişierelor pe disc, sincronizarea fluxurilor şi altele. Cu trecerea timpului, sau făcut progrese în proiectarea lor. O imagine de ansamblu asupra arhitecturii recente a unui astfel de sistem este prezentată în această lucrare.

Tittel et. al, *Mastering XHTML*

Două carţi într-un singur volum, acoperind cel mai nou standard în materie de limbaje de marcaj. Mai întâi este descris limbajul XHTML, cu accent asupra diferenţelor faţă de HTML, apoi este prezentat un ghid cuprinzător al etichetelor, codurilor şi caracterelor speciale utilizate în XHTML 1.0.

Varshney et. al, „Voice over IP”

Cum funcţionează serviciul de voce peste IP şi dacă va înlocui această tehnologie reţeaua comutată de telefonie publică puteţi afla citind această lucrare.

9.1.8 Securitatea reţelelor

Anderson, R., „Why Cryptosystems Fail”

După părerea lui Anderson, securitatea sistemelor bancare este firavă, dar nu datorită intruşilor inteligenţi care sparg DES-ul pe PC-urile lor. Problemele reale sunt foarte variate, de la angajaţi necinstiţi (un angajat de bancă schimbând adresa de poştă a unui client cu a sa, pentru a intercepta numărul cărţii de credit şi numărul de PIN) la erori de programare (acordarea aceluiaşi cod PIN tuturor clienţilor). Este deosebit de interesant răspunsul dat de bancă atunci când se confruntă cu o eroare: sistemul nostru este perfect şi, ca urmare, toate greşelile trebuie să se datoreze erorilor sau fraudelor clienţilor.

Anderson, *Security Engineering*

Într-o anumită măsură, această carte este versiunea de 600 de pagini a lucrării „Why Cryptosystems Fail”. Are o abordare mai tehnică decât *Secrets and Lies* dar mai puţin tehnică decât *Network Security* (descrisă mai jos). După o introducere a tehnicilor de bază de securitate, sunt dedicate capitole întregi diverselor aplicaţii, printre care aplicaţii bancare, comanda şi controlul instalaţiilor nucleare, imprimarea documentelor secrete, biometrie, securitatea fizică, războiul electronic securitatea telecomunicaţiilor, comerţ electronic şi protecţia copyright-ului. A treia parte a lucrării se referă la politici, management şi evoluţia sistemului.

Artz, „Digital Steganography”

Steganografia îşi are originile în Grecia antică, unde ceara era folosită pentru a acoperi inscripţiile făcute pe planşe de lemn, asigurându-se astfel secretul mesajelor. În prezent, sunt utilizate alte tehnici, dar scopul rămâne acelaşi. Există diverse metode de a ascunde informaţia în imagini, fluxuri audio şi alte purtătoare despre care se vorbeşte în această lucrare.

Brands, *Rethinking Public Key Infrastructures and Digital Certificates*

Mai mult decât o introducere cuprinzătoare în utilizarea certificatelor digitale, aceasta este şi o importantă lucrare de avocatură. Autorul are convingerea că documentele de identitate pe suport de hârtie sunt depăşite şi ineficiente şi că certificatele digitale pot fi utilizate pentru aplicaţii ca votul electronic, administrarea drepturilor digitale şi chiar ca înlocuitor pentru bani gheaţă. Autorul atrage atenţia asupra faptului că, dacă nu se foloseşte criptarea PKI, Internetul ar putea deveni un instrument de supraveghere pe scară largă.

Kaufman et. al, *Network Security* ediţia a 2-a

Această carte demnă de încredere şi adesea spirituală este primul loc unde trebuie să căutaţi informaţii despre securitatea reţelelor, despre algoritmii şi protocoalele utilizate. Se explică pe larg şi cu multe exemple algoritmi şi protocoale cu chei secrete şi publice, criptarea mesajelor, autentificarea, Kerberos, PKI, Ipsec, SSL/TSL şi securitatea poştei electronice. Capitolul 26 care tratează subiecte de securitate într-o maniera „folclorică” este o adevarată nestemată. În domeniul securităţii, diavolul se ascunde în detalii. Oricine îşi propune să proiecteze un sistem de securitate care va fi utilizat în practică are ce învăţa din sfaturile

ancorate în realitate prezente în acest capitol.

Pohlmann, *Firewall Systems*

Zidurile de protecţie (firewall) reprezintă pentru cele mai multe reţele prima (şi ultima) linie defensivă împotriva atacatorilor. Această carte descrie cum funcţionează şi ce face un zid de protecţie, de la cele mai simple ziduri de protecţie bazate pe software menite să protejeze un singur PC până la echipamente avansate ce sunt situate între o reţea privată şi conexiunea la Internet.

Schneier, *Applied Criptography*, ediţia a 2-a

Acest compendiu monumental este cel mai mare coşmar al NSA: o singură carte care descrie fiecare algoritm criptografic cunoscut. Pentru a face situaţia şi mai rea (sau mai bună, depinde de punctul dumneavoastră de vedere) cartea conţine cei mai mulţi algoritmi ca programe ce pot fi executate (în C). Mai mult, sunt furnizate peste 1600 de referinţe la literatura criptografică. Deşi nu este pentru începători, dacă vreţi cu adevărat să păstraţi secrete fişierele dumneavoastră, atunci citiţi această carte.

Schneier, *Secrets and Lies*

Dacă citiţi *Applied Criptogyaphy* din scoarţă în scoarţă, veţi afla tot ce este de ştiut referitor la algoritmi criptografici. Dacă mai citiţi apoi şi *Secrets and Lies* din scoarţă în scoarţă (ceea ce nu poate fi făcut într-un timp scurt), veţi afla că algoritmii criptografici nu reprezintă totul. Majoritatea breşelor de securitate nu se datorează unor algoritmi greşiti sau cheilor de criptare prea scurte, ci scurgerilor din mediul securizat. Sunt prezentate exemple fără sfârşit despre ameninţări, atacuri, defensivă, contraatacuri şi multe altele. Pentru o expunere non-tehnică şi fascinantă asupra securităţii computerelor în cel mai larg sens posibil, aceasta este cartea potrivită.

Skoudis, *Counter Hack*

Cel mai bun mod de a opri un hacker este să gândeşti ca un hacker. Această carte descrie viziunea hacker-ilor asupra reţelelor şi argumentează că securitatea ar trebui să fie parte integrantă din proiectarea unei reţele şi nu un adaos bazat pe o anumită tehnologie. Lucrarea acoperă toate tipurile uzuale de atacuri, printre care cele de tip „inginerie socială” care profită de faptul că utilizatorii nu sunt întotdeauna familiari cu măsurile de securitate necesare în utilizarea calculatoarelor.

9.2 BIBLIOGRAFIE ÎN ORDINE ALFABETICĂ

ABRAMSON, N.: “Internet Access Using VSATs”, *IEEE Commun. Magazine,* vol. 38, pag. 60-68, Iulie 2000.

ABRAMSON, N.: “Development of the ALOHANET”, *IEEE Trans. on Information*

Theory, vol. IT- 31, pag. 119-123, Martie 1985.

ADAMS, M. şi DULCHINOS, D.: "OpenCable", *IEEE Commun. Magazine,* vol. 39, pag. 98-105, Iunie 2001.

ALKHATIB, H.S., BAILEY, C., GERLA, M. şi McRAE, J.: "Wireless Data Networks: Reaching the Extra Mile", *Computer,* vol. 30, pag. 59-62, Dec. 1997.

ANDERSON, R.J.: "Free Speech Online and Office", *Computer,* vol. 25, pag. 28-30, Iunie 2002.

ANDERSON, R.J.: *Security Engineering,* New York: Wiley, 2001.

ANDERSON, R.J.: "The Eternity Service", *Proc. First Int'l Conf, on Theory and Appl. of Cryptology,* CTU Publishing House, 1996.

ANDERSON, R.J.: "Why Cryptosystems Fail", *Commun. of the ACM,* vol. 37, pag. 32-40, Nov. 1994.

ARTZ, D.: "Digital Steganography", *IEEE Internet Computing,* vol. 5, pag. 75-80, 2001.

AZZAM, A.A. şi RANSOM, N.: *Broadband Access Technologies,* New York: McGraw-Hill, 1999.

BAKNE, A. şi BADRINATH, B.R.: "I-TCP: Indirect TCP for Mobile Hosts", *Proc. 15th Int'l Conf. on Distr. Computer Systems,* IEEE, pag. 136-143, 1995.

BALAKRISHNAN, H., SESHAN, S. şi KATZ, R.H.: "Improving Reliable Transport and Handoff Performance în Cellular Wireless Networks", *Proc. ACMMobile Computing and Networking Conf.,* ACM, pag. 2-11, 1995.

BALLARDIE, T., FRANCIS, P. şi CROWCROFT, J.: "Core Based Trees (CBT)", *Proc. SIGCOMM '93 Conf.,* ACM, pag. 85-95, 1993.

BARAKAT, C., ALTMAN, E. şi DABBOUS, W.: "On TCP Performance în a Heterogeneous Network: A Survey", *IEEE Commun. Magazine,* vol. 38, pag. 40-46, Ian. 2000.

BELLAMY, J.: *Digital Telephony,* ediţia a treia, New York: Wiley, 2000.

BELLMAN, R.E.: *Dynamic Programming,* Princeton, NJ: Princeton University Press, 1957.

BELSNES, D.: "Flow Control în the Packet Switching Networks", *Communications Networks,* Uxbridge, England: Online, pag. 349-361, 1975.

BENNET, C.H. şi BRASSARD, G.: "Quantum Cryptography: Public Key Distribution and Coin Tossing", *Int'l Conf. on Computer Systems and Signal Processing,* pag. 175-179, 1984.

BEREZDIVIN, R., BREINIG, R. şi TOPP, R.: "Next-Generation Wireless Communication Concepts and Technologies", *IEEE Commun. Magazine,* vol. 40, pag. 108-116, Martie 2002.

BERGHEL, H.L.: "Cyber Privacy în the New Millennium", *Computer,* vol. 34, pag. 132-134, Ian. 2001.

BERGHOLZ, A.: "Extending Your Markup: An XML Tutorial", *IEEE Internet Computing,* vol. 4, pag. 74-79, Iulie-Aug. 2000.

BERNERS-LEE, T., CAILLIAU, A., LOUTONEN, A., NIELSEN, H.F. şi SECRET, A.: "The World Wide Web", *Commun. of the ACM,* vol. 37, pag. 76-82, Aug. 1994.

BERTSEKAS, D. şi GALLAGER, R.: *Data Networks*, ediţia a doua, Englewood Cliffs, NJ: Prentice Hall, 1992.

BHAGWAT, P.: "Bluetooth: Technology for Short-Range Wireless Apps", *IEEE Internet Computin*g, vol. 5, pag. 96-103, Mai-Iunie 2001.

BHARGHAVAN, V., DEMERS, A., SHENKER, S. şi ZHANG, L.: "MACAW: A Media Access Protocol for Wireless LANs", *Proc. SIGCOMM '94 Conf,* ACM, pag. 212-225, 1994.

BHATTI, S.N. şi CROWCROFT, J.: "QoS Sensitive Flows: Issues în IP Packet Handling, *"IEEE Internet Computing*, vol. 4, pag. 48-57, Iulie-Aug. 2000.

BI, Q., ZYSMAN, G.I. şi MENKES, H.: "Wireless Mobile Communications at the Start of the 21st Century", *IEEE Commun. Magazine,* vol. 39, pag. 110-116, Jan, 2001.

BIHAM, E. şi SHAMIR, A.: "Differential Cryptanalysis of the Data Encryption Standard, *"Proc. 17th Ann. Int'l Cryptology Conf.,* Berlin: Springer-Verlag LNCS 1294, pag. 513-525, 1997.

BIRD, R., GOPAL, 1, HERZBERG, A., JANSON, P.A., KUTTEN, S., MOLVA, R şi YUNG, M.: "Systematic Design of a Family of Attack-Resistant Authentication Protocols", *IEEE J. on Selected Areas în Commun.,* vol. 11, pag. 679-693, lunie 1993.

BIRRELL, A.D. şi NELSON, B.J.: "Implementing Remote Procedure Calls", *ACM Trans. on Computer Systems,* vol. 2, pag. 39-59, Feb. 1984.

BIRYUKOV, A., SHAMIR, A. şi WAGNER, D.: "Real Time Cryptanalysis of A5/1 on a PC", *Proc. Seventh Int'l Workshop on Fast Software Encryption,* Berlin: Springer-Verlag LNCS 1978, p. 1, 2000.

BISDIKIAN, C.: "An Overview of the Bluetooth Wireless Technology", *IEEE Commun. Magazine,* vol. 39, pag. 86-94, Dec. 2001.

BLAZE, M.: "Protocol Failure în the Escrowed Encryption Standard", *Proc. Second ACM Conf, on Computer and Commun. Security,* ACM, pag. 59-67, 1994.

BLAZE, M. şi BELLOVIN, S.: "Tapping on My Network Door", *Commun. of the ACM,* vol. 43, pag. 136, Oct. 2000.

BOGINENI, K., SIVALINGAM, KM şi DOWD, P.W.: "Low-Complexity Multiple Access Protocols for Wavelength-Division Multiplexed Photonic Networks", *IEEE Journal on Selected Areas în Commun.,* vol. 11, pag. 590-604, Mai 1993.

BOLCSKEI, H., PAULRAJ, A.J., HARI, K.V.S. şi NABAR, R.U.: "Fixed Broadband Wireless Access: State of the Art, Challenges, and Future Directions", *IEEE Commun. Magazine,* vol. 39, pag. 100-108, Ian. 2001.

BORISOV, N., GOLDBERG, I. şi WAGNER, D.: "Intercepting Mobile Communications: The Insecurity of 802.11", *Seventh Int'l Conf. on Mobile Computing and Networking,* ACM, pag. 180-188, 2001.

BRANDS, S.: *Rethinking Public Key Infrastructures and Digital Certificates,* Cambridge, MA: M.I.T. Press, 2000.

BRAY, J. şi STURMAN, C.F.: *Bluetooth 1.1: Connect without Cables,* editia a doua, Upper Saddle River, NJ: Prentice Hall, 2002.

BREYER, R. şi RILEY, S.: *Switched, Fast, and Gigabit Ethernet,* Indianapolis, IN: New Riders, 1999.

BROWN, S.: *Implementing Virtual Private Networks,* New York: McGraw-Hill, 1999.

BROWN, L., KWAN, M., PIEPRZYK, J. şi SEBERRY, J.: "Improving Resistance to Differential Cryptanalysis and the Redesign of LOKI", *ASIACRYPT '91 Abstracts,* pag. 25-30, 1991.

BURNETT, S. şi PAINE, S.: *RSA Security's Official Guide to Cryptography,* Berkeley, CA: Os- borne/McGraw-Hill, 2001.

CAPETANAKIS, J.I.: "Tree Algorithms for Packet Broadcast Channels", *IEEE Trans. on Information Theory*, vol. IT-25, pag. 505-515, Sept. 1979.

CARDELLINI, V., CASALICCHIO, E., COLAJANNI, M şi YU, P.S.: *"The State-of-the-Art în Locally Distributed Web-Server Systems", ACM Computing Surveys, vol. 34, pag. 263-311, Iunie 2002.*

CARLSON, J.: *PPP Design, Implementation and Debugging,* editia a doua, Boston: Addison- Wesley, 2001.

CERF, V. şi KAHN, R.: "A Protocol for Packet Network Interconnection", *IEEE Trans. on Commun.,* vol. COM-22, pag. 637-648, Mai 1974.

CHAKRABARTI, S.: "QoS Issues în Ad Hoc Wireless Networks", *IEEE Commun. Magazine,* vol. 39, pag. 142-148, Feb. 2001.

CHASE, J.S., GALLATIN, A.J. şi YOCUM, K.G.: "End System Optimizations for High-Speed TCP", *IEEE Commun. Magazine,* vol. 39, pag. 68-75, Aprilie 2001.

CHEN, B., JAMIESON, K., BALAKRISHNAN, H. şi MORRIS, R.: "Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance în Ad Hoc Wireless Net-works, *"ACM Wireless Networks,* vol. 8, Sept. 2002.

CHEN, K.-C.: "Medium Access Control of Wireless LANs for Mobile Computing", *IEEE Network Magazine*, vol. 8, pag. 50-63, Sept.-Oct. 1994.

CHOUDBURY, A.K., MAXEMCHUK, N.F., PAUL, S. şi SCHULZRINNE, H.G.: "Copy-

right Protection for Electronic Publishing on Computer Networks", *IEEE Network Magazine*, vol. 9, pag. 12-20, Mai-Iunie, 1995.

CHU, Y., RAO, S.G. şi ZHANG, H.: "A Case for End System Multicast", *Proc. Int'l Conf, on Measurements and Modeling of Computer Syst.,* ACM, pag. l-12, 2000.

CLARK, D.D.: "The Design Philosophy of the DARPA Internet Protocols", *Proc. SIGCOMM '88 Conf.*, ACM, pag. 106-114, 1988.

CLARK, D.D.: "Window and Acknowledgement Strategy în TCP", RFC 813, lulie 1982.

CLARK, D.D., DAVIE, B.S., FARBER, D.J., GOPAL, I.S., KADABA, B.K., SINCOSKIE, W.D., SMITH, J.M. şi TENNENHOUSE, D.L.: "The Aurora Gigabit Testbed", *Computer Networks and ISDN Systems,* vol. 25, pag. 599-621, Ian. 1993.

CLARK, D.D., JACOBSON, V., ROMKEY, J. şi SALWEN, H.: "An Analysis of TCP Processing Overhead", *IEEE Commun. Magazine,* vol. 27, pag. 23-29, Iunie 1989.

CLARK, D.D., LAMBERT, M. şi ZHANG, L.: "NETBLT: A High Throughput Transport Protocol", *Proc. SIGCOMM '87 Conf,* ACM, pag. 353-359, 1987.

CLARKE, A.C.: "Extra-Terrestrial Relays", *Wireless Worl*d, 1945.

CLARKE, L, MILLER, S.G., HONG, T.W., SANDBERG, O. şi WILEY, B.: "Protecting Free Expression Online with Freenet", *IEEE Internet Computing,* vol. 6, pag. 40-49, Ian.-Feb. 2002.

COLLINS, D.: *Carrier Grade Voice over IP,* New York: McGraw-Hill, 2001.

COLLINS, D. şi SMITH, C.: *3G Wireless Networks,* New York: McGraw-Hill, 2001.

COMER, D.E.: *The Internet Book,* Englewood Cliffs, NJ: Prentice Hall, 1995.

COMER, D.E.: *Internetworking with TCP/IP,* vol. 1, ediţia a patra, Englewood Cliffs, NJ: Pren-tice Hall, 2000.

COSTA, L.H.M.K., FDIDA, S. şi DUARTE, O.C.M.B.: "Hop by Hop Multicast Routing Protocol", *Proc. 2001 Conf. on Applications, Technologies, Architectures, and Proto-cols for Computer Commun.,* ACM, pag. 249-259, 2001.

CRAVER, S.A., WU, M., LIU, B., STUBBLEFIELD, A., SWARTZLANDER, B., WALLACH, D.W., DEAN, D. şi FELTEN, E.W.: "Reading Between the Lines: Lessons from the SDMI Challenge", *Proc. 10h USENIX Security Symp,* USENIX, 2001.

CRESPO, P.M., HONIG, M.L şi SALEHI, J.A.: "Spread-Time Code-Division Multiple Access", *IEEE Trans. on Commun.,* vol. 43, pag. 2139-2148, Iunie 1995.

CROW, B.P., WIDJAJA, I, KIM, J.G. şi SAKAI, P.T.: "IEEE 802.11 Wireless Local Area Networks", *IEEE Commun. Magazine,* vol. 35, pag. 116-126, Sept. 1997

CROWCROFT, J., WANG, Z., SMITH, A. şi ADAMS, J.: "A Rough Comparison of the IETF and ATM Service Models", *IEEE Network Magazine,* vol. 9, pag. 12-16,

Nov.-Dec. 1995.

DABEK, F., BRUNSKILL, E., KAASHOEK, M.F., KARGER, D., MORRIS, R., STOICA, R. şi BALAKRISH- NAN, H.: "Building Peer-to-Peer Systems With Chord, a Distributed Lookup Service", *Proc. 8h Workshop on Hot Topics în Operating Systems,* IEEE, pag. 71-76, 2001a.

DABEK, F., KAASHOEK, M.F., KARGER, D., MORRIS, R. şi STOICA, I.: "Wide-Area Cooperative Storage with CFS", *Proc. 18th Symp. on Operating Systems Prin.,* ACM, pag. 202-15,2001b.

DAEMEN, J. şi RIJMEN, V.: *The Design of Rijndael,* Berlin: Springer-Verlag, 2002.

DANTHINE, A.A.S.: "Protocol Representation with Finite-State Models", *IEEE Trans. on Commun.*, vol. COM-28, pag. 632-643, Aprilie 1980.

DAVIDSON, J. şi PETERS, J.: *Voice over IP Fundamentals,* Indianapolis, IN: Cisco Press, 2000.

DAVIE, B. şi REKHTER, Y.: *MPLS Technology and Applications,* San Francisco: Morgan Kaufmann, 2000.

DAVIS, P.T. şi McGUFFIN, C.R.: *Wireless Local Area Networks,* New York: McGraw-Hill, 1995.

DAVISON, B.D.: "A Web Caching Primer", *IEEE Internet Computing,* vol. 5, pag. 38-45, Iulie-Aug. 2001.

DAY, J.D.: " The (Un)Revised OSI Reference Model", *Computer Commun. Rev.,* vol. 25, pag. 39-55, Oct. 1995.

DAY, J.D. şi ZIMMERMANN, H.: "The OSI Reference Model", *Proc. of the IEEE,* vol. 71, pag. 13341340, Dec. 1983.

DE VRIENDT, J., LAINE, P., LEROUGE, C şi XU, X.: "Mobile Network Evolution: A Revolution on the Move", *IEEE Commun. Magazine,* vol. 40, pag. 104-111, Aprilie 2002.

DEERING, S.E.: "SIP: Simple Internet Protocol", *IEEE Network Magazine,* vol. 7, pag. 16-28, Mai- lunie 1993.

DEMERS, A., KESHAV, S. şi SHENKER, S.: "Analysis and Simulation of a Fair Queueing Algorithm", *Internetwork: Research and Experience,* vol. 1, pag. 3-26, Sept. 1990.

DENNING, D.E. şi SACCO, G.M.: "Timestamps în Key Distribution Protocols", *Commun. of the ACM,* vol. 24, pag. 533-536, Aug. 1981.

DIFFIE, W. şi HELLMAN, M.E.: "Exhaustive Cryptanalysis of the NBS Data Encryption Standard", *Computer*, vol. 10, pag. 74-84, Iunie 1977.

DIFFIE, W. şi HELLMAN, M.E.: "New Directions în Cryptography", *IEEE Trans. on Information Theory*, vol. IT-22, pag. 644-654, Nov. 1976.

DIJKSTRA, E.W.: "A Note on Two Problems în Connexion with Graphs", *Numer. Math.,* vol. 1, pag. 269-271, Oct. 1959.

DOBROWSKI, G. şi GRISE, D.: *ATM and SONET Basics,* Fuquay-Varina, NC: APDG Telecom Books, 2001.

DONALDSON, G. şi JONES, D.: "Cable Television Broadband Network Architectures", *IEEE Commun. Magazine,* vol. 39, pag. 122-126, lunie 2001.

DORFMAN, R.: "Detection of Defective Members of a Large Population", *Annals Math. Statistics,* vol. 14, pag. 436-440, 1943.

DOUFEXI, A., ARMOUR, S., BUTLER, M., NIX, A., BULL, D., MCGEEHAN, J. şi KARLSSON, P.: "A Comparison of the HIPERLAN/2 and IEEE 802.11A Wireless LAN Standards", *IEEE Commun. Magazine,* vol. 40, pag. 172-180, Mai 2002.

DURAND, A.: "Deploying IPv6", *IEEE Internet Computing,* vol. 5, pag. 79-81, Ian.-Feb. 2001.

DUTCHER, B.: *The NAT Handbook,* New York: Wiley, 2001.

DUTTA-ROY, A.: "An Overview of Cable Modem Technology and Market Perspectives", *IEEE Commun. Magazine,* vol. 39, pag. 81-88, Iunie 2001.

EASTTOM, C.: *Learn JavaScript,* Ashburton, U.K.: Wordware Publishing, 2001.

EL GAMAL, T.: "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms", *IEEE Trans. on Information Theory,* vol. IT-31, pag. 469-472, Iulie 1985.

ELHANANY, I., KAHANE, M. şi SADOT, D.: "Packet Scheduling în Next-Generation Multiterabit Networks", *Computer,* vol. 34, pag. 104-106, Aprilie 2001.

ELMIRGHANI, J.M.H. şi MOUFTAH, H.T.: "Technologies and Architectures for Scalable Dynamic Dense WDM Networks", *IEEE Commun. Magazine,* vol. 38, pag. 58-66, Feb. 2000.

FARSEROTU, J. şi PRASAD, R.: "A Survey of Future Broadband Multimedia Satellite Systems, Issues, and Trends", *IEEE Commun. Magazine,* vol. 38, pag. 128-133, Iunie 2000.

FIORINI, D., CHIANI, M., TRALLI, V. şi SALATI., C.: "Can we Trust HDLC?", *Computer Commun. Rev.,* vol. 24, pag. 61-80, Oct. 1994.

FLOYD, S. şi JACOBSON, V.: "Random Early Detection for Congestion Avoidance", *IEEE/ACM Trans. on Networking,* vol. 1, pag. 397-413, Aug. 1993.

FLUHRER, S., MANTIN, I. şi SHAMIR, A.: "Weakness în the Key Scheduling Algorithm of RC4", *Proc. Eighth Ann. Workshop on Selected Areas în Cryptography,* 2001.

FORD, L.R., Jr. şi FULKERSON, D.R.: *Flows în Networks,* Princeton, NJ: Princeton University Press, 1962.

FORD, W. şi BAUM, M.S.: *Secure Electronic Commerce*, Upper Saddle River, NJ: Prentice Hall, 2000.

FORMAN, G.H. şi ZAHORJAN, J.: "The Challenges of Mobile Computing", *Computer,* vol. 27, pag. 3847, Aprilie 1994.

FRANCIS, P.: "A Near-Term Architecture for Deploying Pip", *IEEE Network Magazine*, vol. 7, pag. 30-37, Mai-Iunie 1993.

FRASER, A.G.: "Towards a Universal Data Transport System", în *Advances în Local Area Networks*, Kummerle, K., Tobagi, F. şi Limb, J.O. (Eds.), New York: IEEE Press, 1987.

FRENGLE, N.: *I-Mode: A Primer,* New York: Hungry Minds, 2002.

GADECKI, C. şi HECKERT, C.: *ATM for Dummies,* New York: Hungry Minds, 1997.

GARBER, L.: "Will 3G Really Be the Next Big Wireless Technology?", *Computer,* vol. 35, pag. 26-32, Ian. 2002.

GARFINKEL, S., with SPAFFORD, G.: *Web Security, Privacy, and Commerce*, Sebastopol, CA: O'Reilly, 2002.

GEIER, J.: *Wireless LANs,* ediţia a doua, Indianapolis, IN: Sams, 2002.

GEVROS, P., CROWCROFT, J., KIRSTEIN, P. şi BHATTI, S.: "Congestion Control Mechanisms and the Best Effort Service Model", *IEEE Network Magazine,* vol. 15, pag. 16-25, Mai-Iunie 2001.

GHANI, N. şi DIXIT, S.: "TCP/IP Enhancements for Satellite Networks", *IEEE Commun. Magazine,* vol. 37, pag. 64-72, 1999.

GINSBURG, D.: *ATM: Solutions for Enterprise Networking,* Boston: Addison-Wesley, 1996.

GOODMAN, D.J.: "The Wireless Internet: Promises and Challenges", *Computer*, vol. 33, pag. 36-41, Iulie 2000.

GORALSKI, W.J.: *Optical Networking and WDM,* New York: McGraw-Hill, 2001.

GORALSKI, W.J.: *SONET,* ediţia a doua, New York: McGraw-Hill, 2000.

GORALSKI, W.J.: *Introduction to ATM Networking,* New York: McGraw-Hill, 1995.

GOSSAIN, H., DE MORAIS CORDEIRO şi AGRAWAL, D.P.: "Multicast: Wired to Wireless", *IEEE Commun. Mag.,* vol. 40, pag. 116-123, Iunie 2002.

GRAVANO, S.: *Introduction to Error Control Codes,* Oxford, U.K.: Oxford University Press, 2001.

GUO, Y. şi CHASKAR, H.: "Class-Based Quality of Service over Air Interfaces în 4G Mobile Networks", *IEEE Commun. Magazine,* vol. 40, pag. 132-137, Martie 2002.

HAARTSEN, J.: "The Bluetooth Radio System", *IEEE Personal Commun. Magazine,*

vol. 7, pag. 2836, Feb. 2000.

HAC, A.: "Wireless and Cellular Architecture and Services", *IEEE Commun. Magazine*, vol. 33, pag. 98-104, Nov. 1995.

HAC, A. şi GUO, L.: "A Scalable Mobile Host Protocol for the Internet", *Int'l J. of Network Mgmt,* vol. 10, pag. 115-134, Mai-Iunie, 2000.

HALL, E. şi CERF, V.: *Internet Core Protocols: The Definitive Guide,* Sebastopol, CA: O'Reilly, 2000.

HAMMING, R.W.: "Error Detecting and Error Correcting Codes", *Bell System Tech. J.,* vol. 29, pag. 147-160, Aprilie 1950.

HANEGAN, K.: *Custom CGI Scripting with Perl,* New York: Wiley, 2001.

HARRIS, A.: *JavaScript Programming for the Absolute Beginner,* Premier Press, 2001.

HARTE, L., KELLOGG, S., DREHER, R şi SCHAFFNIT, T.: *The Comprehensive Guide to Wireless Technology,* Fuquay-Varina, NC: APDG Publishing, 2000.

HARTE, L., LEVINE, R. şi KIKTA, R.: *3G Wireless Demystified,* New York: McGraw-Hill, 2002.

HAWLEY, G.T.: "Historical Perspectives on the U.S. Telephone System", *IEEE Commun. Magazine*, vol. 29, pag. 24-28, Martie 1991.

HECHT, J.: "Understanding Fiber Optics", Upper Saddle River, NJ: Prentice Hall, 2001.

HEEGARD, C., COFFEY, J.T., GUMMADI, S., MURPHY, P.A., PROVENCIO, R., ROSSIN, E.J., SCHRUM, S. şi SHOEMAKER, M.B.: "High-Performance Wireless Ethernet", *IEEE Commun. Magazine*, vol. 39, pag. 64-73, Nov. 2001.

HELD, G.: *The Complete Modem Reference,* ediţia a doua, New York: Wiley, 1994.

HELLMAN, M.E.: "A Cryptanalytic Time-Memory Tradeoff", *IEEE Trans. on Information Theor*y, vol. IT-26, pag. 401-406, Iulie 1980.

HILLS, A.: "Large-Scale Wireless LAN Design", *IEEE Commun. Magazine,* vol. 39, pag. 98-104, Nov. 2001.

HOLZMANN, G.J.: *Design and Validation of Computer Protocols,* Englewood Cliffs, NJ: Prentice Hall, 1991.

HU, Y. şi LI, V.O.K.: "Satellite-Based Internet Access", *IEEE Commun. Magazine,* vol. 39, pag. 155162, Martie 2001.

HU, Y.-C. şi JOHNSON, D.B.: "Implicit Source Routes for On-Demand Ad Hoc Network Routing", *Proc. ACMInt'l Symp. on Mobile Ad Hoc Networking & Computing,* ACM, pag. l-10, 2001.

HUANG, V. şi ZHUANG, W.: "QoS-Oriented Access Control for 4G Mobile

Multimedia CDMA Communications", *IEEE Commun. Magazine,* vol. 40, pag. 118-125, Martie 2002.

HUBER, J.F., WEILER, D. şi BRAND, H.: "UMTS, the Mobile Multimedia Vision for IMT-2000: A Focus on Standardization", *IEEE Commun. Magazine,* vol. 38, pag. 129-136, Sept. 2000. nr u 0

HUI, J.: "A Broadband Packet Switch for Multi-rate Services", *Proc. Int'l Conf, on Commun.,* IEEE, pag. 782-788, 1987.

HUITEMA, C.: *Routing în the Internet,* Englewood Cliffs, NJ: Prentice Hall, 1995.

HULL, S.: *Content Delivery Networks,* Berkeley, CA: Osborne/McGraw-Hill, 2002.

HUMBLET, P.A., RAMASWAMI, R şi SIVARAJAN, K.N.: "An Efficient Communication Protocol for High-Speed Packet-Switched Multichannel Networks", *Proc. SIGCOMM '92 Conf.,* ACM, pag. 2-13, 1992.

HUNTER, D.K. şi ANDONOVIC, I.: "Approaches to Optical Internet Packet Switching, " *IEEE Commun. Magazine*, vol. 38, pag. 116-122, Sept. 2000.

HUSTON, G.: "TCP în a Wireless World", *IEEE Internet Computing,* vol. 5, pag. 82-84, Martie- Aprilie, 2001.

IBE, O.C.: *Essentials of ATM Networks and Services,* Boston: Addison-Wesley, 1997.

IRMER, T.: "Shaping Future Telecommunications: The Challenge of Global Standardization, " *IEEE Commun. Magazine,* vol. 32, pag. 20-28, Ian. 1994.

IZZO, P.: *Gigabit Networks,* New York: Wiley, 2000.

JACOBSON, V.: "Congestion Avoidance and Control", *Proc. SIGCOMM '88 Conf.,* ACM, pag. 314329, 1988.

JAIN, R.: "Congestion Control and Traffic Management în ATM Networks: Recent Advances and a Survey", *Computer Networks and ISDN Systems,* vol. 27, Nov. 1995.

JAIN, R.: *FDDI Handbook—High-Speed Networking Using Fiber and Other Media,* Boston: Addison- Wesley, 1994.

JAIN, R.: "Congestion Control în Computer Networks: Issues and Trends", *IEEE Network Magazine,* vol. 4, pag. 24-30, Mai-Iunie 1990.

JAKOBSSON, M. şi WETZEL, S.: "Security Weaknesses în Bluetooth", *Topics în Cryptology: CT-RSA 2001,* Berlin: Springer-Verlag LNCS 2020, pag. 176-191, 2001.

JOEL, A.: "Telecommunications and the IEEE Communications Society", *IEEE Commun. Magazine,* 50th Anniversary Issue, pag. 6-14 and 162-167, Mai 2002.

JOHANSSON, P., KAZANTZIDIS, M., KAPOOR, R şi GERLA, M.: "Bluetooth: An Enabler for Personal Area Networking", *IEEE Network Magazine,* vol. 15, pag. 28-37, Sept.-Oct 2001.

JOHNSON, D.B.: "Scalable Support for Transparent Mobile Host Internetworking", *Wireless Networks*, vol. 1, pag. 311-321, Oct. 1995.

JOHNSON, H.W.: *Fast Ethernet—Dawn of a New Network,* Englewood Cliffs, NJ: Prentice Hall, 1996.

JOHNSON, N.F. şi JAJODA, S.: "Exploring Steganography: Seeing the Unseen", *Computer,* vol. 31, pag. 26-34, Feb. 1998.

KAHN, D.: "Cryptology Goes Public", *IEEE Commun. Magazine,* vol. 18, pag. 19-28, Martie 1980.

KAHN, D.: *The Codebreakers,* ediţia a doua, New York: Macmillan, 1995.

KAMOUN, F. şi KLEINROCK, L.: "Stochastic Performance Evaluation of Hierarchical Routing for Large Networks", *Computer Networks,* vol. 3, pag. 337-353, Nov. 1979.

KAPP, S.: "802.11: Leaving the Wire Behind", *IEEE Internet Computing,* vol. 6, pag. 82- 85, Ian.- Feb. 2002.

KARN, P.: "MACA—A New Channel Access Protocol for Packet Radio", *ARRL/CRRL Amateur Radio Ninth Computer Networking Conf.,* pag. 134-140, 1990.

KARTALOPOULOS, S.: *Introduction to DWDM Technology: Data în a Rainbow,* New York, NY: IEEE Communications Society, 1999.

KASERA, S.K., HJALMTYSSON, G., TOWLSEY, D.F. şi KUROSE, J.F.: "Scalable Reliable Multicast Using Multiple Multicast Channels", *IEEE/ACM Trans. on Networking,* vol. 8, pag. 294-310, 2000.

KATZ, D. şi FORD, P.S.: "TUBA: Replacing IP with CLNP", *IEEE Network Magazine*, vol. 7, pag. 3847, Mai-Iunie 1993.

KATZENBEISSER, S. şi PETITCOLAS, F.A.P.: *Information Hiding Techniques for Steganography and Digital Watermarking,* London, Artech House, 2000.

KAUFMAN, C., PERLMAN, R. şi SPECINER, M.: *Network Security,* ediţia a doua, Englewood Cliffs, NJ: Prentice Hall, 2002.

KELLERER, W., VOGEL, H.-J. şi STEINBERG, K.-E.: "A Communication Gateway for Infrastructure- Independent 4G Wireless Access", *IEEE Commun. Magazine,* vol. 40, pag. 126-131, Martie 2002.

KERCKHOFF, A.: "La Cryptographie Militaire", *J. des Sciences Militaires,* vol. 9, pag. 5- 38, Ian. 1883 şi pag. 161-191, Feb. 1883.

KIM, J.B., SUDA, T. şi YOSHIMURA, M.: "International Standardization of B-ISDN", *Computer Networks and ISDN Systems,* vol. 27, pag. 5-27, Oct. 1994.

KIPNIS, J.: "Beating the System: Abuses of the Standards Adoptions Process", *IEEE Commun. Magazine,* vol. 38, pag. 102-105, lulie 2000.

KLEINROCK, L.: "On Some Principles of Nomadic Computing and Multi-Access

Communications, *"IEEE Commun. Magazine,* vol. 38, pag. 46-50, lulie 2000.

KLEINROCK, L. şi TOBAGI, F.: "Random Access Techniques for Data Transmission over Packet- Switched Radio Channels", *Proc. Nat. Computer Conf.,* pag. 187-201, 1975.

KRISHNAMURTHY, B. şi REXFORD, J.: *Web Protocols and Practice,* Boston: Addison-Wesley, 2001.

KUMAR, V., KORPI, M. şi SENGODAN, S.: *IP Telephony with H.323,* New York: Wiley, 2001.

KUROSE, J.F. şi ROSS, K.W.: *Computer Networking: A Top-Down Approach Featuring the Internet,* Boston: Addison-Wesley, 2001.

KWOK, T.: "A Vision for Residential Broadband Service: ATM to the Home", *IEEE Network Magazine*, vol. 9, pag. 14-28, Sept.-Oct. 1995.

KYAS, O. şi CRAWFORD, G.: *ATM Networks,* Upper Saddle River, NJ: Prentice Hall, 2002.

LAM, C.K.M. şi TAN, B.C.Y.: "The Internet Is Changing the Music Industry", *Commun. of the ACM,* vol. 44, pag. 62-66, Aug. 2001.

LANSFORD, J., STEPHENS, A şi NEVO, R.: "Wi-Fi (802.11b) and Bluetooth: Enabling Coexistence", *IEEE Network Magazine,* vol. 15, pag. 20-27, Sept.-Oct 2001.

LASH, D.A.: *The Web Wizard's Guide to Perl and CGI,* Boston: Addison-Wesley, 2002.

LAUBACH, M.E., FARBER, D.J. şi DUKES, S.D.: *Delivering Internet Connections over Cable,* New York: Wiley, 2001.

LEE, J.S. şi MILLER, L.E.: *CDMA Systems Engineering Handbook,* London: Artech House, 1998.

LEEPER, D.G.: "A Long-Term View of Short-Range Wireless", *Computer,* vol. 34, pag. 39-44, Iunie 2001.

LEINER, B.M., COLE, R., POSTEL, J. şi MILLS, D.: "The DARPA Internet Protocol Suite", *IEEE Commun. Magazine*, vol. 23, pag. 29-34, Martie 1985.

LEVINE, D.A. şi AKYILDIZ, LA.: "PROTON: A Media Access Control Protocol for Optical Networks with Star Topology", *IEEE/ACM Trans. on Networking,* vol. 3, pag. 158-168, Aprilie 1995.

LEVY, S.: "Crypto Rebels", *Wired,* pag. 54-61, Mai-Iunie 1993.

LI, J., BLAKE, C., DE COUTO, D.S.J., LEE, H.I. şi MORRIS, R.: "Capacity of Ad Hoc Wireless Networks", *Proc. 7th Int'l Conf. on Mobile Computing and Networking,* ACM, pag. 61-69, 2001.

LIN, F., CHU, P. şi LIU, M.: "Protocol Verification Using Reachability Analysis: The State Space Explosion Problem and Relief Strategies", *Proc. SIGCOMM '87*

Conf., ACM, pag. 126-135, 1987.

LIN, Y.-D., HSU, N.-B. şi HWANG, R.-H.: "QoS Routing Granularity în MPLS Networks", *IEEE Commun. Magazine,* vol. 40, pag. 58-65, lunie 2002.

LISTANI, M., ERAMO, V. şi SABELLA, R.: "Architectural and Technological Issues for Future Optical Internet Networks", *IEEE Commun. Magazine,* vol. 38, pag. 82-92, Sept. 2000.

LIU, C.L. şi LAYLAND, J.W.: "Scheduling Algorithms for Multiprogramming în a Hard Real-Time Environment", *Journal of the ACM,* vol. 20, pag. 46-61, Ian. 1973.

LIVINGSTON, D.: *Essential XML for Web Professionals,* Upper Saddle River, NJ: Prentice Hall, 2002.

LOSHIN, P.: *IPv6 Clearly Explained,* San Francisco: Morgan Kaufmann, 1999.

LOUIS, P.J.: *Broadband Crash Course,* New York: McGraw-Hill, 2002.

LU, W.: *Broadband Wireless Mobile: 3G and Beyond,* New York: Wiley, 2002.

MACEDONIA, M.R.: "Distributed File Sharing", *Computer,* vol. 33, pag. 99-101, 2000.

MADRUGA, E.L. şi GARCIA-LUNA-ACEVES, J.J.: "Scalable Multicasting: the Core-Assisted Mesh Protocol", *Mobile Networks and Applications,* vol. 6, pag. 151-165, Aprilie 2001.

MALHOTRA, R.: *IP Routing,* Sebastopol, CA: O'Reilly, 2002.

MATSUI, M.: "Linear Cryptanalysis Method for DES Cipher", *Advances în Cryptology— Eurocrypt '93 Proceedings,* Berlin: Springer-Verlag LNCS 765, pag. 386-397, 1994.

MAUFER, T.A.: *IP Fundamentals,* Upper Saddle River, NJ: Prentice Hall, 1999.

MAZIERES, D. şi KAASHOEK, M.F.: "The Design, Implementation, and Operation of an Email Pseudonym Server", *Proc. Fifth Conf, on Computer and Commun. Security,* ACM, pag. 27-36, 1998.

MAZIERES, D., KAMINSKY, M., KAASHOEK, M.F. şi WITCHEL, E.: "Separating Key Management from File System Security", *Proc. 17th Symp. on Operating Systems Prin.,* ACM, pag. 124-139, Dec. 1999.

McFEDRIES, P: *Using JavaScript,* Indianapolis, IN: Que, 2001.

McKENNEY, P.E. şi DOVE, K.F.: "Efficient Demultiplexing of Incoming TCP Packets, " *Proc. SIGCOMM '92 Conf.,* ACM, pag. 269-279, 1992.

MELTZER, K. şi MICHALSKI, B.: *Writing CGI Applications with Perl,* Boston: Addison-Wesley, 2001.

MENEZES, A.J. şi VANSTONE, S.A.: "Elliptic Curve Cryptosystems and Their

Implementation, " *Journal of Cryptology,* vol. 6, pag. 209-224, 1993.

MERKLE, R.C.: "Fast Software Encryption Functions", *Advances în Cryptology—CRYPTO '90 Proceedings,* Berlin: Springer-Verlag LNCS 473, pag. 476-501, 1991.

MERKLE, R.C. şi HELLMAN, M.: "On the Security of Multiple Encryption", *Commun. of the AC*M, vol. 24, pag. 465-467, Iulie 1981.

MERKLE, R.C. şi HELLMAN, M.: "Hiding and Signatures în Trapdoor Knapsacks", *IEEE Trans. on Information Theory,* vol. IT-24, pag. 525-530, Sept. 1978.

METCALFE, R.M.: "On Mobile Computing", *Byte,* vol. 20, p. 110, Sept. 1995.

METCALFE, R.M.: "Computer/Network Interface Design: Lessons from Arpanet and Ethernet, " *IEEE Journal on Selected Areas în Commun.,* vol. 11, pag. 173-179, Feb. 1993.

METCALFE, R.M. şi BOGGS, D.R.: "Ethernet: Distributed Packet Switching for Local Computer Networks", *Commun. of the ACM,* vol. 19, pag. 395-404, lulie 1976.

METZ, C: "Interconnecting ISP Networks", *IEEE Internet Computing*, vol. 5, pag. 74-80, Martie- Aprilie 2001.

METZ, C.: "Differentiated Services", *IEEE Multimedia Magazine,* vol. 7, pag. 84-90, lulie-Sept. 2000.

METZ, C.: "IP Routers: New Tool for Gigabit Networking", *IEEE Internet Computing,* vol. 2, pag. 1418, Nov.-Dec. 1998.

MILLER, B.A. şi BISDIKIAN, C.,: *Bluetooth Revealed,* Upper Saddle River, NJ: Prentice Hall, 2001.

MILLER, P. şi CUMMINS, M.: *LAN Technologies Explained,* Woburn, MA: Butterworth-Heine – mann, 2000.

MINOLI, D.: *Video Dialtone Technology,* New York: McGraw-Hill, 1995.

MINOLI, D. şi VITELLA, M.: *ATM & Cell Relay for Corporate Environments,* New York: McGrawHill, 1994.

MISHRA, P.P. şi KANAKIA, H.: "A Hop by Hop Rate-Based Congestion Control Scheme", *Proc. SIGCOMM '92 Conf,* ACM, pag. 112-123, 1992.

MISRA, A., DAS, S., DUTTA, A., MCAULEY, A. şi DAS, S.: "IDMP-Based Fast Handoffs and Paging în IP-Based 4G Mobile Networks", *IEEE Commun. Magazine,* vol. 40, pag. 138-145, Martie 2002.

MOGUL, J.C.: "IP Network Performance", în *Internet System Handbook,* Lynch, D.C. and Rose, M.T. (eds.), Boston: Addison-Wesley, pag. 575-675, 1993.

MOK, A.K. şi WARD, S.A.: "Distributed Broadcast Channel Access", *Computer Networks,* vol. 3, pag. 327-335, Nov. 1979.

MOY, J.: "Multicast Routing Extensions", *Commun. of the ACM,* vol. 37, pag. 61-66, Aug. 1994.

MULLINS, J.: "Making Unbreakable Code", *IEEE Spectrum,* pag. 40-45, Mai 2002.

NAGLE, J.: "On Packet Switches with Infinite Storage", *IEEE Trans. on Commun.*, vol. COM-35, pag. 435-438, Aprilie 1987.

NAGLE, J.: "Congestion Control în TCP/IP Internetworks", *Computer Commun. Rev.*, vol. 14, pag. 11-17, Oct. 1984.

NARAYANASWAMI, C., KAMIJOH, N., RAGHUNATH, M., INOUE, T., CIPOLLA, T., SANFORD, J., SCHLIG, E., VENTKITESWARAN, S., GUNIGUNTALA, D., KULKARNI, V. şi YAMAZAKI, K.: "IBM's Linux Watch: The Challenge of Miniaturization", *Computer,* vol. 35, pag. 33-41, Ian. 2002.

NAUGHTON, J.: "A Brief History of the Future", Woodstock, NY: Overlook Press, 2000.

NEEDHAM, R.M. şi SCHROEDER, M.D.: "Authentication Revisited", *Operating Systems Rev.,* vol. 21, p. 7, Ian. 1987.

NEEDHAM, R.M. şi SCHROEDER, M.D.: "Using Encryption for Authentication în Large Networks of Computers", *Commun. of the ACM,* vol. 21, pag. 993-999, Dec. 1978.

NELAKUDITI, S. şi ZHANG, Z.-L.: "A Localized Adaptive Proportioning Approach to QoS Routing", *IEEE Commun. Magazine* vol. 40, pag. 66-71, Iunie 2002.

NEMETH, E., SNYDER, G., SEEBASS, S. şi HEIN, T.R.: *UNIX System Administration Handbook,* ediţia a treia, Englewood Cliffs, NJ: Prentice Hall, 2000.

NICHOLS, R.K. şi LEKKAS, P.C.: *Wireless Security,* New York: McGraw-Hill, 2002.

NIST: "Secure Hash Algorithm", U.S. Government Federal Information Processing Standard 180, 1993.

O'HARA, B. şi PETRICK, A.: *802.11 Handbook: A Designer's Companion,* New York: IEEE Press, 1999.

OTWAY, D. şi REES, O.: "Efficient and Timely Mutual Authentication", *Operating Systems Rev.,* pag. 8-10, Ian. 1987.

OVADIA, S.: *Broadband Cable TV Access Networks: from Technologies to Applications,* Upper Saddle River, NJ: Prentice Hall, 2001.

PALAIS, J.C.: *Fiber Optic Commun.,* ediţia a treia, Englewood Cliffs, NJ: Prentice Hall, 1992.

PAN, D.: "A Tutorial on MPEG/Audio Compression", *IEEE Multimedia Magazine,* vol. 2, pag.60-74, Summer 1995.

PANDYA, R.: "Emerging Mobile and Personal Communication Systems", *IEEE*

Commun. Magazine, vol. 33, pag. 44-52, Iunie 1995.

PARAMESWARAN, M., SUSARLA, A. şi WHINSTON, A.B.: "P2P Networking: An Information-Sharing Alternative", *Computer,* vol. 34, pag. 31-38, Iulie 2001.

PARK, J.S. şi SANDHU, R.: "Secure Cookies on the Web", *IEEE Internet Computing,* vol. 4, pag. 36-44, Iulie-Aug. 2000.

PARTRIDGE, C., HUGHES, J. şi STONE, J.: "Performance of Checksums and CRCs over Real Data", *Proc. SIGCOMM '95 Conf.,* ACM, pag. 68-76, 1995.

PAXSON, V.: "Growth Trends în Wide-Area TCP Connections", *IEEE Network Magazine,* vol. 8, pag. 8-17, Iulie-Aug. 1994.

PAXSON, V. şi FLOYD, S.: "Wide-Area Traffic: The Failure of Poisson Modeling", *Proc. SIGCOMM '94 Conf,* ACM, pag. 257-268, 1995.

PEPELNJAK, I. şi GUICHARD, J.: *MPLS and VPN Architectures,* Indianapolis, IN: Cisco Press, 2001.

PERKINS, C.E.: *RTP: Audio and Video for the Internet,* Boston: Addison-Wesley, 2002.

PERKINS, C.E. (ed.): *Ad Hoc Networking,* Boston: Addison-Wesley, 2001.

PERKINS, C.E.: *Mobile IP Design Principles and Practices,* Upper Saddle River, NJ: Prentice Hall, 1998a.

PERKINS, C.E.: "Mobile Networking în the Internet", *Mobile Networks and Applications,* vol. 3, pag. 319-334, 1998b.

PERKINS, C.E.: "Mobile Networking through Mobile IP", *IEEE Internet Computing,* vol. 2, pag. 5869, Ian.-Feb. 1998c.

PERKINS, C.E. şi ROYER, E.: "The Ad Hoc On-Demand Distance-Vector Protocol", în *Ad Hoc Networking,* edited by C. Perkins, Boston: Addison-Wesley, 2001.

PERKINS, C.E. şi ROYER, E.: "Ad-hoc On-Demand Distance Vector Routing", *Proc. Second Ann. IEEE Workshop on Mobile Computing Systems and Applications,* IEEE, pag. 90-100, 1999.

PERLMAN, R.: *Interconnections,* editia a doua, Boston: Addison-Wesley, 2000.

PERLMAN, R.: *Network Layer Protocols with Byzantine Robustness,* Ph.D. thesis, M.I.T., 1988.

PERLMAN, R. şi KAUFMAN, C.: "Key Exchange în IPsec", *IEEE Internet Computing,* vol. 4, pag. 5056, Nov.-Dec. 2000.

PETERSON, L.L. şi DAVIE, B.S.: *Computer Networks: A Systems Approach,* San Francisco: Morgan Kaufmann, 2000.

PETERSON, W.W. şi BROWN, D.T.: "Cyclic Codes for Error Detection", *Proc. IRE,* vol. 49, pag. 228235, Ian. 1961.

PICKHOLTZ, R.L., SCHILLING, D.L. şi MILSTEIN, L.B.: "Theory of Spread Spectrum Communication—A Tutorial", *IEEE Trans. onCommun.,* vol. COM-30, pag. 855-884, Mai 1982.

PIERRE, G., KUZ, I., VAN STEEN, M., TANENBAUM, A.S.: "Differentiated Strategies for Replicating Web Documents", *Computer Commun.,* vol. 24, pag. 232-240, Feb. 2001.

PIERRE, G., VAN STEEN, M. şi TANENBAUM, A.S.: "Dynamically Selecting Optimal Distribution Strategies for Web Documents", *IEEE Trans. on Computers,* vol. 51, Iunie 2002.

PISCITELLO, D.M. şi CHAPIN, A.L.: *Open Systems Networking: TCP/IP and OSI,* Boston: Addison- Wesley, 1993.

PITT, D.A.: "Bridging—The Double Standard", *IEEE Network Magazine,* vol. 2, pag. 94- 95, Ian. 1988.

PIVA, A., BARTOLINI, F. şi BARNI, M.: "Managing Copyrights în Open Networks", *IEEE Internet Computing,* vol. 6, pag. 18-26, Mai-Iunie 2002.

POHLMANN, N.: *Firewall Systems,* Bonn, Germany: MITP-Verlag, 2001.

PUZMANOVA, R.: *Routing and Switching: Time of Convergence?,* London: Addison-Wesley, 2002.

RABINOVICH, M. şi SPATSCHECK, O,: *Web Caching and Replication,* Boston: Addison-Wesley, 2002.

RAJU, J. şi GARCIA-LUNA-ACEVES, J.J.: "Scenario-based Comparison of Source-Tracing and Dynamic Source Routing Protocols for Ad-Hoc Networks", *ACM Computer Communications Review,* vol. 31, October 2001.

RAMANATHAN, R. şi REDI, J.: "A Brief Overview of Ad Hoc Networks: Challenges and Directions", *IEEE Commun. Magazine,* 50th Anniversary Issue, pag. 20-22, Mai 2002.

RATNASAMY, S., FRANCIS, P., HANDLEY, M., KARP, R şi SHENKER, S.: "A Scalable Content- Addressable Network", *Proc. SIGCOMM '01 Conf,* ACM, pag. 1161-1172, 2001.

RIVEST, R.L.: "The MD5 Message-Digest Algorithm", RFC 1320, Aprilie 1992.

RIVEST, R.L. şi SHAMIR, A.: "How to Expose an Eavesdropper", *Commun. of the ACM,* vol. 27, pag. 393-395, Aprilie 1984.

RIVEST, R.L., SHAMIR, A. şi ADLEMAN, L.: "On a Method for Obtaining Digital Signatures and Public Key Cryptosystems", *Commun. of the ACM,* vol. 21, pag. 120-126, Feb. 1978.

ROBERTS, L.G.: "Dynamic Allocation of Satellite Capacity through Packet Reservation", *Proc. NCC,* AFIPS, pag. 711-716, 1973.

ROBERTS, L.G.: "Extensions of Packet Communication Technology to a Hand Held

Personal Terminal", *Proc. Spring Joint Computer Conference,* AFIPS, pag. 295-298, 1972.

ROBERTS, L.G.: "Multiple Computer Networks and Intercomputer Communication", *Proc. First Symp. on Operating Systems Prin,* ACM, 1967.

ROSE, M.T.: *The Simple Book,* Englewood Cliffs, NJ: Prentice Hall, 1994.

ROSE, M.T.: *The Internet Message,* Englewood Cliffs, NJ: Prentice Hall, 1993.

ROSE, M.T. şi MCCLOGHRIE, K.: *How to Manage Your Network Using SNMP,* Englewood Cliffs, NJ: Prentice Hall, 1995.

ROWSTRON, A. şi DRUSCHEL, P.: "Storage Management and Caching în PAST, a Large-Scale, Persistent Peer-to-Peer Storage Utility", *Proc. 18h Symp. on Operating Systems Prin.,* ACM, pag. 188-201, 2001a.

ROWSTRON, A. şi DRUSCHEL, P.: "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Storage Utility", *Proc. 18h Int'l Conf. on Distributed Systems Platforms,* ACM/IFIP, 2001b.

ROYER, E.M. şi TOH, C.-K.: "A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks", *IEEE Personal Commun. Magazine,* vol. 6, pag. 46-55, Aprilie 1999.

RUIZ-SANCHEZ, M.A.[7], BIERSACK, E.W. şi DABBOUS, W.: "Survey and Taxonomy of IP Address Lookup Algorithms", *IEEE Network Magazine,* vol. 15, pag. 8-23, Martie-Aprilie 2001.

SAIRAM, K.V.S.S.S.S., GUNASEKARAN, N. şi REDDY, S.R.: "Bluetooth în Wireless Communication", *IEEE Commun. Mag.,* vol. 40, pag. 90-96, Iunie 2002.

SALTZER, J.H., REED, D.P. şi CLARK, D.D.: "End-to-End Arguments în System Design", *ACM Trans. on Computer Systems,* vol. 2, pag. 277-288, Nov. 1984.

SANDERSON, D.W. şi DOUGHERTY, D.: *Smileys,* Sebastopol, CA: O'Reilly, 1993.

SARI, H., VANHAVERBEKE, F. şi MOENECLAEY, M.: "Extending the Capacity of Multiple Access Channels", *IEEE Commun. Magazine,* vol. 38, pag. 74-82, Ian. 2000.

SARIKAYA, B.: "Packet Mode în Wireless Networks: Overview of Transition to Third Generation", *IEEE Commun. Magazine,* vol. 38, pag. 164-172, Sept. 2000.

SCHNEIER, B.: *Secrets and Lies,* New York: Wiley, 2000.

SCHNEIER, B.: *Applied Cryptography,* ediţia a doua, New York: Wiley, 1996.

SCHNEIER, B.: *E-Mail Security,* New York: Wiley, 1995.

SCHNEIER, B.: "Description of a New Variable-Length Key, 64-Bit Block Cipher [Blowfish]", *Proc. of the Cambridge Security Workshop,* Berlin: Springer-Verlag LNCS 809, pag. 191-204, 1994.

SCHNORR, C.P.: "Efficient Signature Generation for Smart Cards", *Journal of Cryptology,* vol. 4, pag. 161-174, 1991.

SCHOLTZ, R.A.: "The Origins of Spread-Spectrum Communications", *IEEE Trans. on Commun.*, vol. COM-30, pag. 822-854, Mai 1982.

SCOTT, R.: "Wide Open Encryption Design Offers Flexible Implementations", *Cryptologia,* vol. 9, pag. 75-90, Ian. 1985.

SEIFERT, R.: *The Switch Book,* Boston: Addison-Wesley, 2000.

SEIFERT, R.: *Gigabit Ethernet,* Boston: Addison-Wesley, 1998.

SENN, J.A.: "The Emergence of M-Commerce", *Computer,* vol. 33, pag. 148-150, Dec. 2000.

SERJANTOV, A.: "Anonymizing Censorship Resistant Systems", *Proc. First Int'l Workshop on Peer-to- Peer Systems,* Berlin: Springer-Verlag LNCS, 2002.

SEVERANCE, C.: "IEEE 802.11: Wireless Is Coming Home", *Computer,* vol. 32, pag. 126-127, Nov. 1999.

SHAHABI, C., ZIMMERMANN, R., FU, K. şi YAO, S.-Y.D.: "YIMA: A Second-

Generation Content Networks", *IEEE Computer*, vol. 35, pag. 56-64, Iunie 2002.

SHANNON, C.: "A Mathematical Theory of Communication", *Bell System Tech. J.*, vol. 27, pag. 379423, Iulie 1948; şi pag. 623-656, Oct. 1948.

SHEPARD, S.: *SONET/SDH Demystified,* New York: McGraw-Hill, 2001.

SHREEDHAR, M. şi VARGHESE, G.: "Efficient Fair Queueing Using Deficit Round Robin", *Proc. SIGCOMM '95 Conf,* ACM, pag. 231-243, 1995.

SKOUDIS, E.: *Counter Hack,* Upper Saddle River, NJ: Prentice Hall, 2002.

SMITH, D.K. şi ALEXANDER, R.C.: *Fumbling the Future,* New York: William Morrow, 1988.

SMITH, R.W.: *Broadband Internet Connections,* Boston: Addison Wesley, 2002.

SNOEREN, A.C. şi BALAKRISHNAN, H.: "An End-to-End Approach to Host Mobility, " *Int'l Conf. on Mobile Computing and Networking* , ACM, pag. 155-166, 2000.

SOBEL, D.L.: "Will Carnivore Devour Online Privacy", *Computer,* vol. 34, pag. 87-88, Mai 2001.

SOLOMON, J.D.: *Mobile IP: The Internet Unplugged,* Upper Saddle River, NJ: Prentice Hall, 1998.

SPOHN, M. şi GARCIA-LUNA-ACEVES, J.J.: "Neighborhood Aware Source Routing", *Proc. ACM Mo- biHoc 2001,* ACM, pag. 2001.

SPURGEON, C.E.: *Ethernet: The Definitive Guide,* Sebastopol, CA: O'Reilly, 2000.

STALLINGS, W.: *Data and Computer Communications,* ediţia a şasea, Upper Saddle River, NJ: Prentice Hall, 2000.

STEINMETZ, R. şi NAHRSTEDT, K.: *Multimedia Fundamentals. Vol. 1: Media Coding and Content Processing*, Upper Saddle River, NJ: Prentice Hall, 2002.

STEINMETZ, R. şi NAHRSTEDT, K.: *Multimedia Fundamentals. Vol. 2: Media Processing and Communications*, Upper Saddle River, NJ: Prentice Hall, 2003a.

STEINMETZ, R. şi NAHRSTEDT, K.: *Multimedia Fundamentals. Vol. 3: Documents, Security, and Applications*, Upper Saddle River, NJ: Prentice Hall, 2003b.

STEINER, J.G., NEUMAN, B.C. şi SCHILLER, J.I.: "Kerberos: An Authentication Service for Open Network Systems", *Proc. Winter USENIX Conf.,* USENIX, pag. 191- 201, 1988.

STEVENS, W.R.: *UNIX Network Programming, Volume 1: Networking APIs – Sockets and XTI,* Upper Saddle River, NJ: Prentice Hall, 1997.

STEVENS, W.R.: *TCP/IP Illustrated,* Vol. 1, Boston: Addison-Wesley, 1994.

STEWART, R. şi METZ, C.: "SCTP: New Transport Protocol for TCP/IP", *IEEE Internet Computing,* vol. 5, pag. 64-69, Nov.-Dec. 2001.

STINSON, D.R.: *Cryptography Theory and Practice,* ediţia a doua, Boca Raton, FL: CRC Press, 2002.

STOICA, I., MORRIS, R., KARGER, D., KAASHOEK, MF. şi BALAKRISHNAN, H.: "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications", *Proc.*

SIGCOMM '01 Conf., ACM, pag. 149-160, 2001.

STRIEGEL, A. şi MANIMARAN, G.: "A Survey of QoS Multicasting Issues", *IEEECommun. Mag.,* vol. 40, pag. 82-87, lunie 2002.

STUBBLEFIELD, A., IOANNIDIS, J. şi RUBIN, A.D.: "Using the Fluhrer, Mantin şi Shamir Attack to Break WEP", *Proc Network and Distributed Systems Security Symp.,* ISOC, pag. l-11, 2002.

SUMMERS, C.K.: *ADSL: Standards, Implementation, and Architecture,* Boca Raton, FL: CRC Press, 1999.

SUNSHINE, C.A. şi DALAL, Y.K.: "Connection Management în Transport Protocols", *Computer Networks,* vol. 2, pag. 454-473, 1978.

TANENBAUM, A.S.: *Modern Operating Systems,* Upper Saddle River, NJ: Prentice Hall, 2001.

TANENBAUM, A.S. şi VAN STEEN, M.: *Distributed Systems: Principles and Paradigms,* Upper Saddle River, NJ: Prentice Hall, 2002.

TEGER, S. şi WAKS, D.J.: "End-User Perspectives on Home Networking", *IEEE Commun. Magazine,* vol. 40, pag. 114-119, Aprilie 2002.

THYAGARAJAN, A.S. şi DEERING, S.E.: "Hierarchical Distance-Vector Multicast Routing for the MBone", *Proc. SIGCOMM '95 Conf.,* ACM, pag. 60-66, 1995.

TTTTEL, E., VALENTINE, C., BURMEISTER, M şi DYKES, L.: *Mastering XHTML,* Alameda, CA: Sybex, 2001.

TOKORO, M. şi TAMARU, K.: "Acknowledging Ethernet", *Compcon,* IEEE, pag. 320-325, Fall 1977.

TOMLINSON, R.S.: "Selecting Sequence Numbers", *Proc. SIGCOMM/SIGOPS Interprocess Commun. Workshop,* ACM, pag. 11-23, 1975.

TSENG, Y.-C., WU, S.-L., LIAO, W.-H. şi CHAO, C.-M.: "Location Awareness în Ad Hoc Wireless Mobile Networks", *Computer*, vol. 34, pag. 46-51, 2001.

TUCHMAN, W.: "Hellman Presents No Shortcut Solutions to DES", *IEEE Spectrum,* vol. 16, pag. 4041, lulie 1979.

TURNER, J.S.: "New Directions în Communications (or Which Way to the Information Age)", *IEEE Commun. Magazine,* vol. 24, pag. 8-15, Oct. 1986.

VACCA, J.R.: *I-Mode Crash Course,* New York: McGraw-Hill, 2002.

VALADE, J.,: *PHP & MySQL for Dummies,* New York: Hungry Minds, 2002.

VARGHESE, G. şi LAUCK, T.: "Hashed and Hierarchical Timing Wheels: Data Structures for the Efficient Implementation of a Timer Facility", *Proc. 11th Symp. on Operating Systems Prin.,* ACM, pag. 25-38, 1987.

VARSHNEY, U., SNOW, A., M**CGIVERN, M. şi HOWARD, C.:** "Voice over IP",

Commun. of the ACM, vol. 45, pag. 89-96, 2002.

VARSHNEY, U. şi VETTER, R.: "Emerging Mobile and Wireless Networks", *Commit. of the ACM,* vol. 43, pag. 73-81, lunie 2000.

VETTER, P., GODERIS, D., VERPOOTEN, L. şi GRANGER, A.: "Systems Aspects of APON/VDSL Deployment", *IEEECommun. Magazine,* vol. 38, pag. 66-72, Mai 2000.

WADDINGTON, D.G. şi CHANG, F.: "Realizing the Transition to IPv6", *IEEE Commun. Mag.,* vol. 40, pag. 138-148, lunie 2002.

WALDMAN, M., RUBIN, A.D. şi CRANOR, L.F.: "Publius: A Robust, Tamper-Evident, CensorshipResistant, Web Publishing System", *Proc. Ninth USENIX Security Symp.,* USENIX, pag. 5972, 2000.

WANG, Y. şi CHEN, W.: "Supporting IP Multicast for Mobile Hosts", *Mobile Networks and Applications,* vol. 6, pag. 57-66, lan.-Feb. 2001.

WANG, Z.: *Internet QoS,* San Francisco: Morgan Kaufmann, 2001.

WARNEKE, B., LAST, M., LIEBOWITZ, B. şi PISTER, K.S.J.: "Smart Dust: Communicating with a Cubic Millimeter Computer", *Compute*r, vol. 34, pag. 44-51, Ian. 2001.

WAYNER, P.: *Disappearing Cryptography: Information Hiding, Steganography, and Watermarking,* editia a doua, San Francisco: Morgan Kaufmann, 2002.

WEBB, W.: "Broadband Fixed Wireless Access as a Key Component of the Future Integrated Communications Environment", *IEEE Commun. Magazine,* vol. 39, pag. 115-121, Sept. 2001.

WEISER, M.: "Whatever Happened to the Next Generation Internet?", *Commun. of the AC*M, vol. 44, pag. 61-68, Sept. 2001.

WELTMAN, R. şi DAHBURA, T.: *LDAP Programming with Java,* Boston: Addison-Wesley, 2000.

WESSELS, D.: *Web Caching,* Sebastopol, CA: O'Reilly, 2001.

WETTEROTH, D.: *OSI Reference Model for Telecommunications,* New York: McGraw-Hill, 2001.

WILJAKKA, J.: "Transition to Ipv6 în GPRS and WCDMA Mobile Networks", *IEEE Commun. Magazine,* vol. 40, pag. 134-140, Aprilie 2002.

WILLIAMSON, H.:*XML: The Complete Reference,* New York: McGraw-Hill, 2001.

WILLINGER, W., TAQQU, M.S., SHERMAN, R şi WILSON, D.V.: "Self-Similarity through High Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level", *Proc. SIGCOMM '95 Conf.*, ACM, pag. 100-113, 1995.

WRIGHT, D.J.: *Voice over Packet Networks,* New York: Wiley, 2001.

WYLIE, J., BIGRIGG, M.W., STRUNK, J.D., GANGER, G.R., KILICCOTE, H. şi KHOSLA,

P.K.: "Survivable Information Storage Systems", *Computer,* vol. 33, pag. 61-68, Aug. 2000.

XYLOMENOS, G., POLYZOS, G.C., MAHONEN, P. şi SAARANEN, M.: "TCP Performance Issues over ■ ■ ■ ■ ■ , ■ Wireless Links" , *IEEE Commun. Magazine,* vol. 39, pag. 52-58, Aprilie 2001.

YANG, C.-Q. şi REDDY, A.V.S.: "A Taxonomy for Congestion Control Algorithms în Packet Switching Networks", *IEEE Network Magazine,* vol. 9, pag. 34-45, lulie-Aug. 1995.

YUVAL, G.: "How to Swindle Rabin", *Cryptologia,* vol. 3, pag. 187-190, lulie 1979.

ZACKS, M.: "Antiterrorist Legislation Expands Electronic Snooping", *IEEE Internet Computing,* vol. 5, pag. 8-9, Nov.-Dec. 2001.

ZADEH, A.N., JABBARI, B., PICKHOLTZ, R. şi VOJCIC, B.: "Self-Organizing Packet Radio Ad Hoc Networks with Overlay (SOPRANO)", *IEEE Commun. Mag.,* vol. 40, pag. 149-157, lunie 2002.

ZHANG, L.: "Comparison of Two Bridge Routing Approaches", *IEEE Network Magazine,* vol. 2, pag. 44-48, Ian.-Feb. 1988.

ZHANG, L.: "RSVP: A New Resource ReSerVation Protocol", *IEEE Network Magazine,* vol. 7, pag. 8-18, Sept.-Oct. 1993.

ZHANG, Y. şi RYU, B.: "Mobile and Multicast IP Services în PACS: System Architecture, Prototype, and Performance", *Mobile Networks and Applications,* vol. 6, pag. 81- 94, Ian.-Feb. 2001.

ZIMMERMANN, P.R.: *The Official PGP User's Guide,* Cambridge, MA: M.I.T. Press, 1995a.

ZIMMERMANN, P.R.: *PGP: Source Code and Internals,* Cambridge, MA: M.I.T. Press, 1995b.

ZIPF, G.K.: *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology,* Boston: Addison-Wesley, 1949.

ZIV, J. şi LEMPEL, Z.: "A Universal Algorithm for Sequential Data Compression", *IEEE Trans. on Information Theory,* vol. IT-23, pag. 337-343, Mai 1977.

INDEX

AAL, 57, 58, 442
ADC, 603
ALOHA, 226, 227, 228, 229, 230, 231, 232, 233,
236, 302, 303, 304
AMPS, 138, 140, 159
ANSI, 66, 67, 209
ANSNET, 50
ARP, 402, 404, 405, 414, 423, 428, 429
ARPANET, 37, 47, 48, 49, 50, 68, 84, 322, 324, 391,
511, 518, 522, 530, 536
ARQ, 187
ATM, 44, 57, 58, 71, 130, 311, 360, 376, 383, 384,
426, 429, 442, 457, 461, 499, 500

B

BGP, 411, 412, 416, 423
BOC, 110
BOOTP, 402, 405

C

CDPD, 329
CIDR, 395, 396, 415
Consiliul Arhitecturii Internet, 68
CSMA, 231, 232, 233, 235, 242, 243, 251, 254, 302, 304, 305
CSMA/CD, 232, 233, 254, 302, 304, 305

D

DES, 660, 661, 662, 666, 667, 668, 671, 674, 679,
711, 716
DNS, 39, 48, 404, 416, 521, 522, 524, 525, 526, 527,
529, 533, 534, 551, 552, 558, 559
domeniu, 7, 24, 41, 67, 113, 124, 125, 160, 218, 225,
241, 242, 254, 315, 335, 336, 416, 522,
523, 524, 525, 526, 528, 571, 648, 711 DSS, 677

F

FAQ, 547
FCC, 97, 103, 138
FDDI, 403, 405
FDM, 101, 123, 124, 125, 126, 140, 159, 163, 224,
225, 238, 302
FTP, 39, 51, 478, 539, 559, 560, 694

G

Gopher, 559, 560
GSM, 304

H

HDLC, 70, 209, 211, 212, 214, 215, 217, 219, 220
HDTV, 620, 626
HTML, 538, 559, 564, 565, 566, 567, 568, 569, 570, 599
HTTP, 40, 559, 579, 580, 581, 582, 584,

585, 586, 588, 589, 592, 614, 616
DCS,

I

IAB, 68, 69

IBM, 16, 43, 50, 64, 83, 209, 411, 660, 661, 662 ICMP, 402, 413, 416, 419, 420, 421, 423, 493 IDEA, 714, 715
IEEE, 16, 68, 69, 242, 243, 248, 288, 289, 416, 662 IEEE 802, 16, 242, 243, 288
IGMP, 413, 416 IMP, 46, 47, 49
IMTS, 138
IP, 34, 37, 38, 39, 40, 41, 42, 43, 44, 48, 50, 57, 68, 72, 74, 213, 214, 216, 220, 329, 375, 376, 380, 381, 383, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 402, 403, 404, 405, 407, 410, 412, 413, 414, 415, 416, 417, 418, 419, 422, 423, 425, 427, 428, 429, 442, 471, 477, 478, 479, 480, 481, 482, 484, 488, 495, 496, 503, 508, 509, 516, 522, 524, 525, 526, 527, 528, 551, 559, 694
IPv4, 415, 416, 417, 418, 419, 421, 422, 429 IPX, 214, 329, 375, 376
ISDN, 130
IS-IS, 329
ISO, 34, 35, 66, 67, 68, 69, 209, 329, 522, 566, 622 ITU, 65, 66, 68, 91, 622
ITU-R, 65
ITU-T, 65, 66, 68 IXC, 110, 111

J

JPEG, 538, 567, 622, 623, 624, 625, 626, 627

K

KDC, 701, 707, 708, 709, 710
Kerberos, 709, 710, 711

L

LAN, 16, 18, 21, 24, 48, 73, 89, 212, 223, 224, 225, 226, 227, 232, 238, 241, 242, 249, 250, 253, 254, 286, 287, 288, 289, 290, 291, 292, 302, 304, 305, 311, 325, 329, 335, 392, 403, 409, 413, 414, 425, 429
MAC, 223, 224, 232, 233, 241, 248, 287, 292, 383
MACA, 242, 243, 302, 304
MACAW, 242, 243, 302
MAN, 16, 224, 225, 335, 518
MD5, 678, 679, 682, 714, 715, 716
MIME, 536, 537, 538, 585, 594, 595, 597, 598, 600, 601, 602, 610, 714
Mosaic, 51, 548, 549
MPEG, 538, 539, 622, 625, 626
MTSO, 139, 140
MTU, 480

N

NAK, 203, 219
NAP, 50
NCP, 213, 214, 215, 216, 375
NIC, 391
NIST, 68, 677, 678, 679
NNTP, 39, 560
Novell, 329, 375
NSA, 661, 662, 677, 679
NSAP, 442, 443
NSFNET, 49, 50, 52, 329
NTSC. 619. 620. 623.

OAM, 130
OC, 132
OSPF, 329, 406, 407, 408, 409, 410, 411, 413, 416, 423

P

PAL, 619, 620, 623, 626
PCM, 126, 127, 130, 161
PCS, 163
PEM, 712, 716
PGP, 712, 713, 714, 715, 716
POP, 110
POP3, 543
PPP, 70, 213, 214, 215, 217, 220
PSTN, 107
PTT, 64

Q

QAM, 115

R

RFC, 69, 214, 216, 390, 396, 404, 405, 406, 407, 412, 413, 416, 419, 471, 477, 483, 534, 535, 536, 537, 538, 539, 543, 714, 716
RSVP, 368

S

SABME, 211
SAP, 74, 442
SDH, 129, 131, 132
SDLC, 209, 211, 217
SECAM, 619, 620, 623
SGML, 538
SHA, 678, 679
SIPP, 416
SLIP, 213
Smiley, 529
SMTP, 39, 540, 541, 542, 543
SNA, 209
SNRME, 211
SONET, 44, 124, 129, 130, 131, 132, 162, 214, 215, 621
SPE, 131, 132

T

TCP, 34, 37, 38, 39, 40, 41, 42, 43, 44, 48, 50, 57, 68, 72, 74, 213, 375, 379, 381, 390, 412, 416, 418, 419, 436, 444, 471, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 499, 500, 503, 508, 509, 514, 517, 522, 540, 541, 542, 551, 559, 694 TDM, 124, 129, 130, 159, 163, 225, 238, 302, 383 TPDU, 434, 435, 436, 437, 445, 446, 447, 448, 449, 450, 451, 453, 454, 455, 456, 457, 458, 460, 499, 500, 502, 503, 505, 507, 508, 509, 517
TSAP, 442, 443, 444, 445, 460, 461, 478, 481

U

UDP, 39, 74, 239, 390, 405, 416, 418, 419, 471, 496, 497, 500, 503, 516, 522, 528, 694
URI, 560
URL, 551, 558, 559, 560, 567, 585
UTP, 83

V

V.24, 65
V.32, 116
VSAT, 101, 102

W

WAN, 17, 23, 24, 168, 223, 311, 335, 381, 405, 432
WDM, 125, 126, 159
WDMA, 238, 240, 304
Web, 40, 51, 503, 521, 538, 548, 549, 550, 552, 558, 559, 560, 564, 565, 567, 568, 569, 571, 584, 585, 586, 617, 694
WWW, 51, 179, 497, 548, 551, 559, 565

X

X.400, 530, 533

DESPRE AUTOR

Andrew S. Tanenbaum este absolvent al M.I.T. şi a obţinut titlul de doctor la University of California din Berkeley. În acest moment este profesor la Vrije Universiteit din Amsterdam, Olanda, unde este conducător al grupului care se ocupă de sistemele de calculatoare (Computer Systems Group). De asemenea, este decan al Şcolii Avansate de Calculatoare şi Prelucrarea Imaginilor (Advanced School for Computing and Imaging), un institut de învăţământ interuniversitar care se ocupă cu studiul sistemelor paralele, distribuite şi de prelucrare a imaginilor avansate. Cu toate acestea, el încearcă din răsputeri să evite să devină un birocrat.

În trecut, a efectuat cercetări în domeniul compilatoarelor, al sistemelor de operare, al reţelelor şi al sistemelor distribuite locale. Domeniul în care efectuează acum cercetări este cel al sistemelor distribuite globale care pot fi scalate până la un miliard de utilizatori. Aceste cercetări, efectuate împreună cu prof. Maarten van Steen sunt prezentate la *www.cs.vu.nl/globe*. Împreună, toate aceste proiecte de cercetare au dus la scrierea a peste 100 de articole publicate în diverse reviste sau prezentate la diferite conferinţe şi la scrierea a cinci cărţi.

Prof. Tanenbaum a creat, de asemenea, numeroase produse software. A fost principalul arhitect al produsului Amsterdam Compiler Kit, un set de unelte folosit pe scară largă pentru scrierea de compilatoare portabile. De asemenea, este creatorul sistemului de operare MINIX, o clonă UNIX destinată utilizării la orele de laborator ale studenţilor. Acest sistem de operare a fost sursa de inspiraţie şi baza sistemului de operare LINUX. Împreună cu doctoranzii săi şi cu mai mulţi programatori, a contribuit la proiectarea sistemului de operare distribuit Amoeba, un sistem foarte performant bazat pe un micronucleu. MINIX şi Amoeba sunt acum disponibile gratuit pe Internet.

Doctoranzii săi şi-au urmat drumul spre glorie după obţinerea titlurilor de doctori. E foarte mândru de ei. În această privinţă, el se aseamănă cu o cloşcă.

Prof. Tanenbaum este membru ACM, IEEE, precum şi al Academiei Regale de Arte şi Ştiinţe din Olanda (Royal Netherlands Academy of Arts and Sciences). De-a lungul timpului a obţinut numeroase distincţii: ACM Karl V. Karlstrom Outstanding Educator Award în 1994, ACM/SIGCSE Award for Outstanding Contributions to Computer Science Education în 1997 şi Texty în 2002. Aceasta din urmă este acordată pentru excelenţă în scrierea cărţilor. De asemenea, numele său şi o scurtă prezentare apare în cartea *Who's Who în the World*. Pagina sa Web este disponibilă la adresa *http://www.cs.vu.nl/~ast/*.